CSC258 - Breakout Project Report

Mani Setayesh, Leon Cai

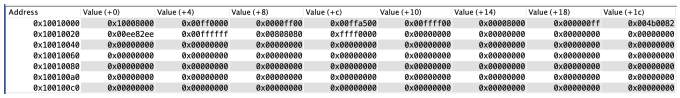
November 28, 2022

Memory Layout

Things that need to be laid out:

- A ball - it should have (x, y) values, in addition to the "direction" of the ball. The direction of the ball follows an encoding - e.g. if a value of 1 is stored, then direction is left, etc.
- A paddle - it only needs (x, y) values (frankly not even the y value since the paddle only moves horizontally).
- Colours - just an array of the colours used in the display. Multiple colours (one for each row), and 3 monochrome colours (black,gray,white) stored at the end- used for the ball, the paddle, the walls, and empty space.

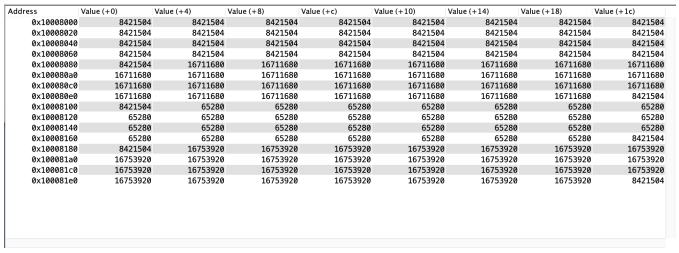
Memory layout at the beginning:



Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x10008000	0x00ff0000	0x0000ff00	0x00ffa500	0x00ffff00	0x00808000	0x000000ff	0x004b0082
0x10010020	0x00ee82ee	0x00ffffff	0x00808080	0xffff0000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Figure 1: Initial memory

The values in the memory include the keyboard display address (ADDR_DSPL) - given already in the file. Then the array of 10 colours, then spaces left for the x, y values of the ball and paddle. It is hard to get the memory's layout for the walls and bricks in one picture, as what we will have is:



Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10008000	8421504	8421504	8421504	8421504	8421504	8421504	8421504	8421504
0x10008020	8421504	8421504	8421504	8421504	8421504	8421504	8421504	8421504
0x10008040	8421504	8421504	8421504	8421504	8421504	8421504	8421504	8421504
0x10008060	8421504	8421504	8421504	8421504	8421504	8421504	8421504	8421504
0x10008080	8421504	16711680	16711680	16711680	16711680	16711680	16711680	16711680
0x100080a0	16711680	16711680	16711680	16711680	16711680	16711680	16711680	16711680
0x100080c0	16711680	16711680	16711680	16711680	16711680	16711680	16711680	16711680
0x100080e0	16711680	16711680	16711680	16711680	16711680	16711680	16711680	8421504
0x10008100	8421504	65280	65280	65280	65280	65280	65280	65280
0x10008120	65280	65280	65280	65280	65280	65280	65280	65280
0x10008140	65280	65280	65280	65280	65280	65280	65280	65280
0x10008160	65280	65280	65280	65280	65280	65280	65280	8421504
0x10008180	8421504	16753920	16753920	16753920	16753920	16753920	16753920	16753920
0x100081a0	16753920	16753920	16753920	16753920	16753920	16753920	16753920	16753920
0x100081c0	16753920	16753920	16753920	16753920	16753920	16753920	16753920	16753920
0x100081e0	16753920	16753920	16753920	16753920	16753920	16753920	16753920	8421504

Figure 2: Coloured memory

Notice that the repeated numbers each indicate a row from the top - the top row is coloured gray, done by storing "gray" values into 32 word spaces (each representing a pixel). Then a row of red, then green, then etc. Here is a screen-shot of the static scene (at Statthe start of the game):

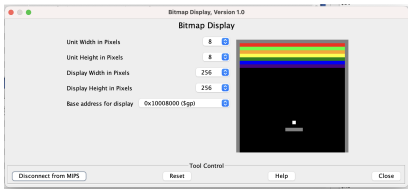


Figure 3: Static scene

Dealing with collisions

The main aspect with dealing with a collision is changing the already stored "direction" value (gotten from address: BALL + 8) to a new direction that the ball should go in. Using classic breakout as reference, the ball mainly moves in diagonals. We associate each diagonal with a number from 1-4: Up-right = 1, Up-left = 2, Down-left = 3, Down-right = 4 (cntr clkwise from top-right). The general idea is for the ball's direction to rotate 90 degrees when it hits anything - though the direction of the rotation should be intuitive:

- Ball hits the paddle - in this case, the ball should only change whether its going up/down, not left/right. For example, if it hits the paddle with dir = 3, then the new dir would be 2. If it hits the paddle with dir = 4, then the new dir would be 1.
- Ball hits a wall: Differentiating top walls from side walls, we have the direction that should change and that should not change. For the side wall collision, the ball should keep its vertical direction - but it should change its horizontal direction (if going left, hit the left wall, then go right). For the top wall, its the other way around - keep the horizontal direction, but instead of going up the ball will go down. If it hits a corner, then the new direction is the opposite of the old direction (going up-right, hit corner, go down-left).
- Ball hits a brick - the same idea as hitting the top wall.

Here is a visual diagram for how it the bouncing directions would work:

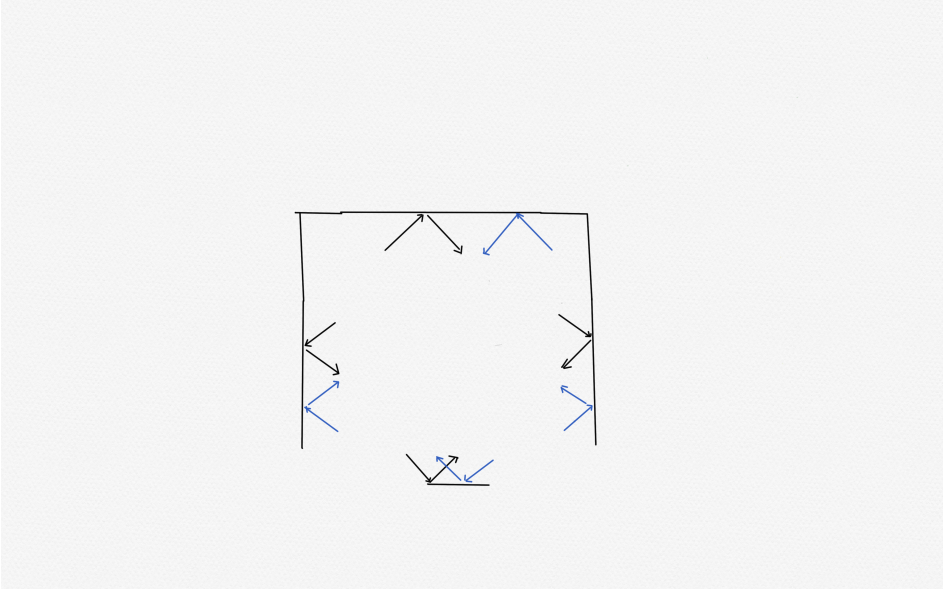


Figure 4: Collisions, relative to wall/paddle