

Breakout project report

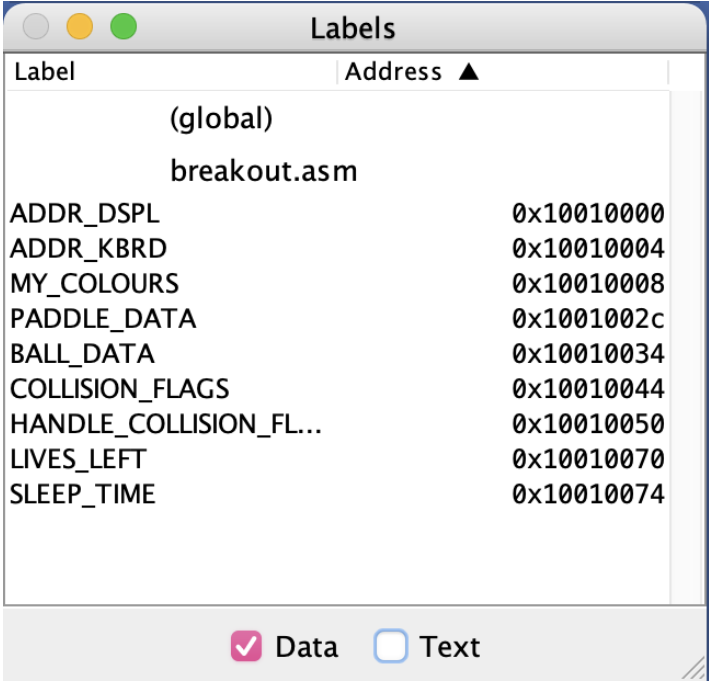
Hongshuo Zhou, Mengyuan Li

December 6, 2022

1 Memory allocation

In our project, we decide to store :

- the address of the bitmap display(immutable data)
- the address of the keyboard(immutable data)
- the colours used to draw lines and pixels(immutable data)
- the coordinates of our data(mutable data)
- the coordinates and moving direction of our ball(mutable data)
- Collision Flags that used to record if there is collision happened.
- handle collision flags
- LIVES_LEFT* which is how many lives user have left before losing the game
- SLEEP_TIME* which is the sleep time of each cycle



The screenshot shows a window titled 'Labels' with a table of memory addresses. The table has two columns: 'Label' and 'Address'. The labels are listed under the heading '(global) breakout.asm'. The addresses are in hexadecimal format. At the bottom of the window, there are two checkboxes: 'Data' (checked) and 'Text' (unchecked).

Label	Address ▲
(global)	
breakout.asm	
ADDR_DSPL	0x10010000
ADDR_KBRD	0x10010004
MY_COLOURS	0x10010008
PADDLE_DATA	0x1001002c
BALL_DATA	0x10010034
COLLISION_FLAGS	0x10010044
HANDLE_COLLISION_FL...	0x10010050
LIVES_LEFT	0x10010070
SLEEP_TIME	0x10010074

Figure 1: An image of memory

2 Static Scene

3 How will the ball change directions when it collides?

We use movement coordinates to represent the direction the ball moves which is stored in *BALL_DATA* the memory section.

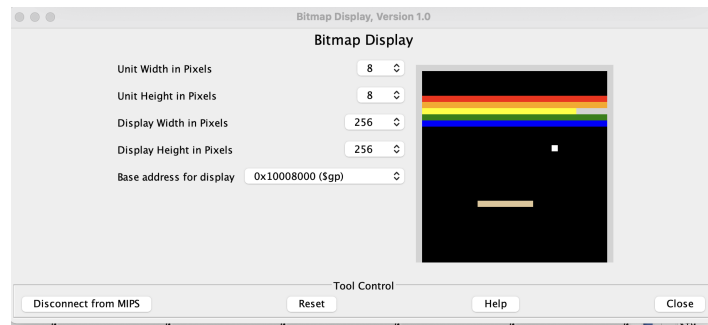


Figure 2: An image of our static scene(initialized screen)

Our setting is when $x = 0$, ball moves left, when $x = 1$ ball moves right; when $y = 0$, ball moves down, when $y = 1$, ball moves up.

we can split all situations into following cases(the pic is shown below):

Case_one: The only obstacle appears on the *top* of the ball. Now movement coordinates of y changed from 1 to 0, x unchanged.

Case_two: The only obstacle appears on the *bottom* of the ball. Now movement coordinates of y changed from 1 to 0, x unchanged.

Case_three: The only obstacle appears on the *left* of the ball, the collision occurred on the left side of it. Now movement coordinates of x toggled where y remains the same.

Case_four: Obstacles appeared on *top* and *left* of the ball. Now movement coordinates of both x and y toggled(from 0 to 1, 1 to 0).

Case_five: The only obstacle appears on the *diagonal* of the ball. Now movement coordinates of both x and y toggled.

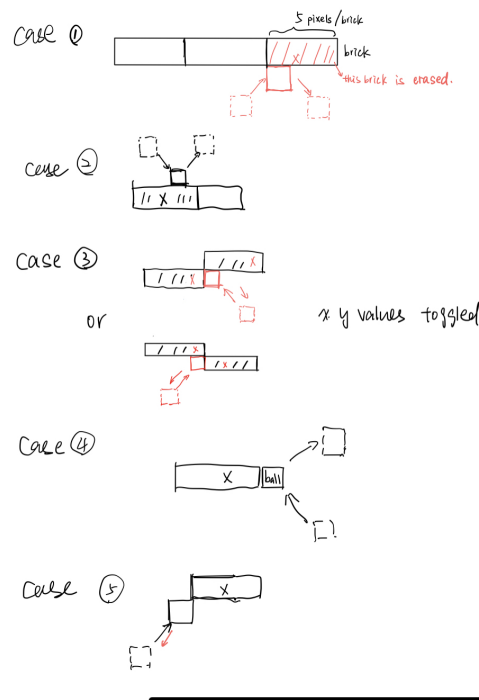


Figure 3: A demonstration of above cases

4 HOW TO PLAY OUR GAME:

Set up instructions:

In bitmap display, choose 8 for unit width and unit height in pixels. Choose 256X256 for display size. Connect both bitmap display and keyboard.

1. In order to win the game, users need to break all of the bricks off except for the grey unbreakable brick before losing all 3 lives. Bricks need to be broken by ball for twice, in the first collision, the brick will turn red, in the second collision, the brick will disappear. Users need to use keyboard input *a* and *d* to control the movement of the paddle moving left and moving right in order to bounce the ball into any direction they want. Users will notice the speed of ball movement also changes when collision happens.

2. When the ball falls on the ground instead of being held by the paddle, user will lose one life, after losing all 3 lives, users will see a yellow game over screen and they can choose to press *r* to restart or *q* to quit the game.

3. When user wants to pause the game, pressing *p* and when user is ready to restart the game, they can start exactly at the place they left by pressing *p* again.