BÁO CÁO BÀI TẬP PROJECT CUỐI KÌ

MÔN THỰC HÀNH KIẾN TRÚC MÁY TÍNH

Giảng viên hướng dẫn: Đỗ Công Thuần

Sinh viên thực hiện:

Nguyễn Đức Long - 20225876 2

PROJECT : BITMAP

PHỤ TRÁCH: NGUYỄN ĐỨC LONG - 20225876

1) REQUIREMENTS

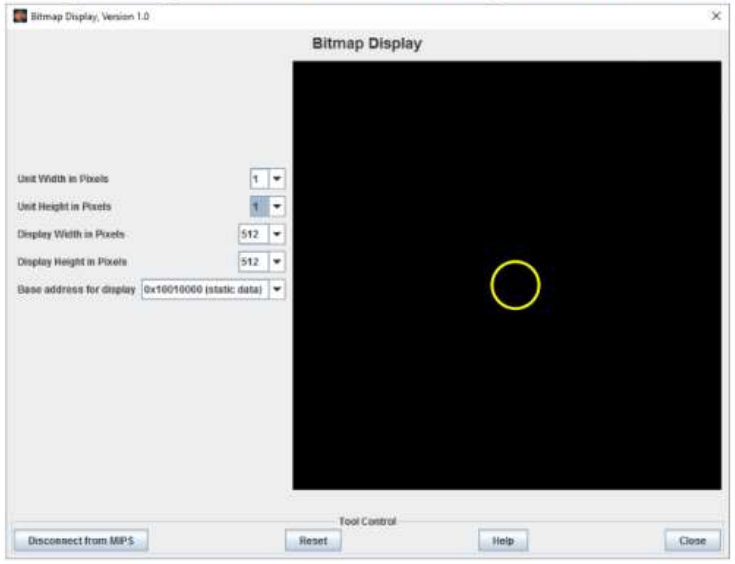
Viết một chương trình sử dụng MIPS để vẽ một quả bóng di chuyển trên màn hình mô phỏng Bitmap của Mars). Nếu đối tượng đập vào cạnh của màn hình thì sẽ di chuyển theo chiều ngược lại.

Yêu cầu:

- Thiết lập màn hình ở kích thước 512x512. Kích thước pixel 1x1.
- Quả bóng là một đường tròn.

Chiều di chuyển phụ thuộc vào phím người dùng bấm, gồm có (di chuyển lên (W), di chuyển xuống (S), Sang trái (A), Sang phải (D) trong bộ giả lập Keyboard and Display MMIO Simulator). Vị trí bóng ban đầu ở giữa màn hình. Tốc độ bóng di chuyển là có thay đổi không đổi. Khi người dùng giữ một phím nào đó (W, S, A, D) thì quả bóng sẽ tăng tốc theo hướng đó với gia tốc tuỳ chọn.

Gợi ý: *Để làm một đối tượng di chuyển thì chúng ta sẽ xóa đối tượng ở vị trí cũ và vẽ đối tượng ở vị trí mới. Để xóa đối tượng chúng ta chỉ cần vẽ đối tượng đó với màu là màu nền.*



1) REQUIREMENTS

- - Draw circle in the center of screen, (NO MOVE)
- - PRESS ‘WASD’ to go up, left, down, right
- - Press the button and hold (= press multiple times) to speed up:

  for example: hold (press multiple times) W to go up with highspeed
- - If circle hit any edge of screen, change the direction!

2) OVERVIEW OF THE PROCESS

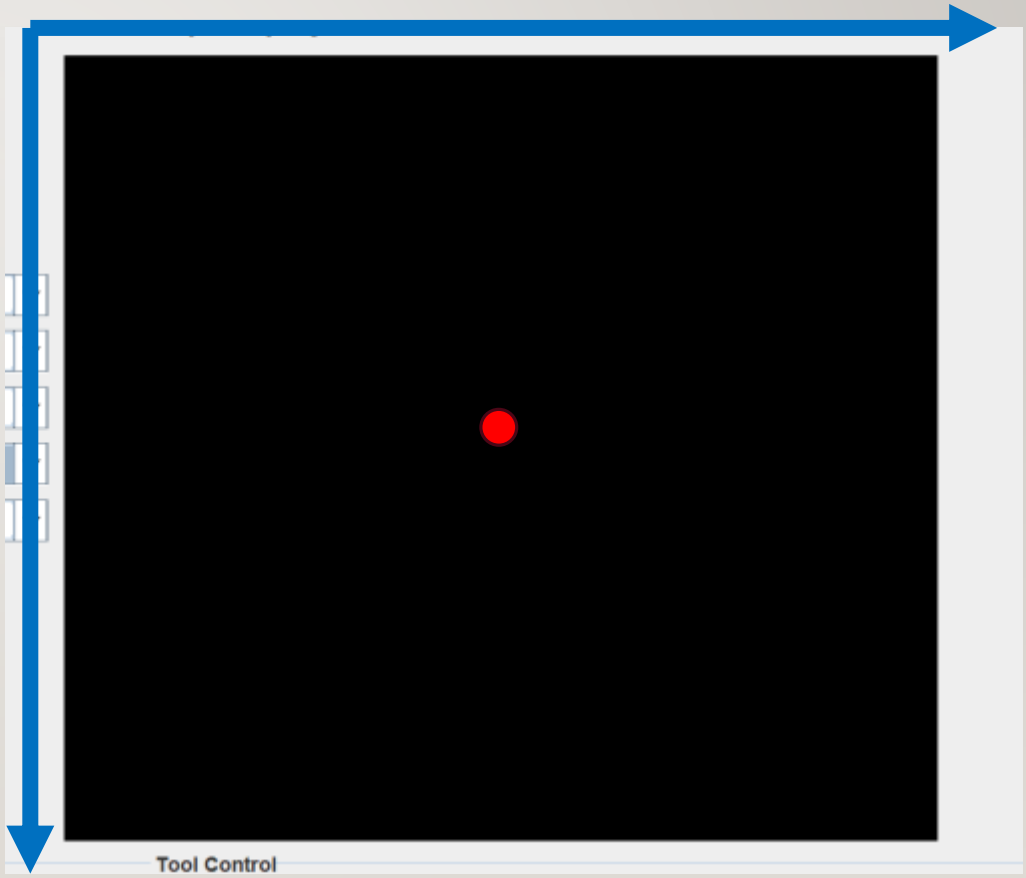
- S1: Init some basic values: circle center, radius, dx dy movement distance corresponding to movement direction (ex: dx = -1, dy = 0 is going left or dx = 0, dy = 1 is going down), $a0 contains sleep time (sleep less -> move faster).
- S2: Calculate the points to create the circle (calculate the position of the points with the reference frame relative to the center of the circle), then save all these points into an array.
- S3:Read from input and check if there is any change in direction/speed. If there is no change then it is still moving as before, proceed with edge touching check.

2) OVERVIEW OF THE PROCESS

- S4: Check if the circle has touched the edge of the screen. If it has, change the direction of movement (and draw the circle in a new location). If not, continue moving in the same direction (and draw the circle in a new location).
- S5: Delete the previous circle (Use black color, paint over the previous data (current data) of the circle, then update the new data of the circle, use yellow color, paint over these new positions
- S6: Repeat the process from step 3.

3) DETAILS OF ALL STEPS
STEP1: SETUP CIRCLE CENTER DATA

```
setup:
    li $s0, 255      # x = 255
    li $s1, 255      # y = 255
    li $s2, 0        # dx = 0
    li $s3, 0        # dy = 0
    li $s4, 20       # r = 20
    li $a0, 40       # t = 40ms
    jal      get_circle_data
```



3) DETAILS OF ALL STEPS
PHẦN 2: CALCULATE THE RELATIVE POSITIONS OF POINTS TO FORM A CIRCLE

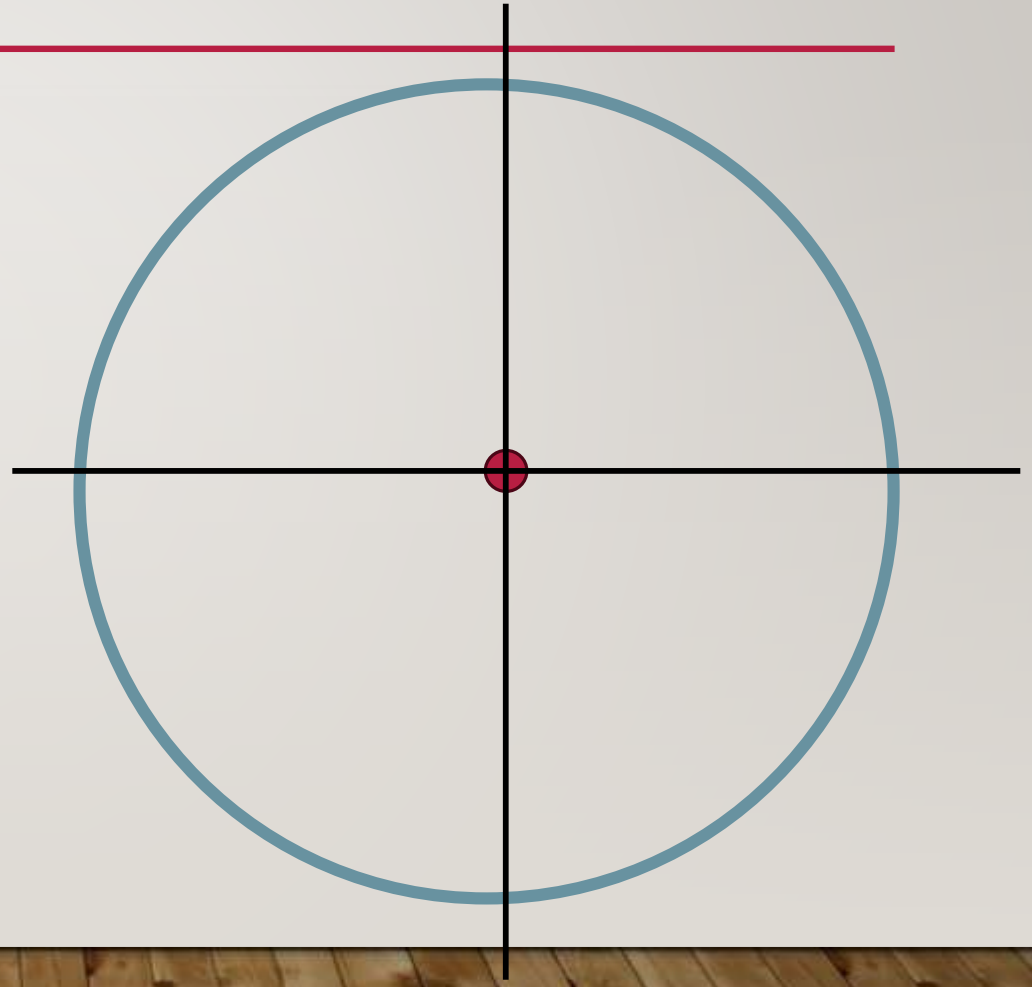
- Find the point with coordinates (px, py) that satisfy the circle by:
- Find all points that have the distance to center equal to (20) by this formula:

  -> 20^2 = px^2 + py^2

  -> py = sqrt(400 – px^2)

  (px from: 0 -> 20)

- So we can find all points (px, py) that make up the circle



3) DETAILS OF ALL STEPS
PHẦN 2: CALCULATE THE RELATIVE POSITIONS OF POINTS TO FORM A CIRCLE

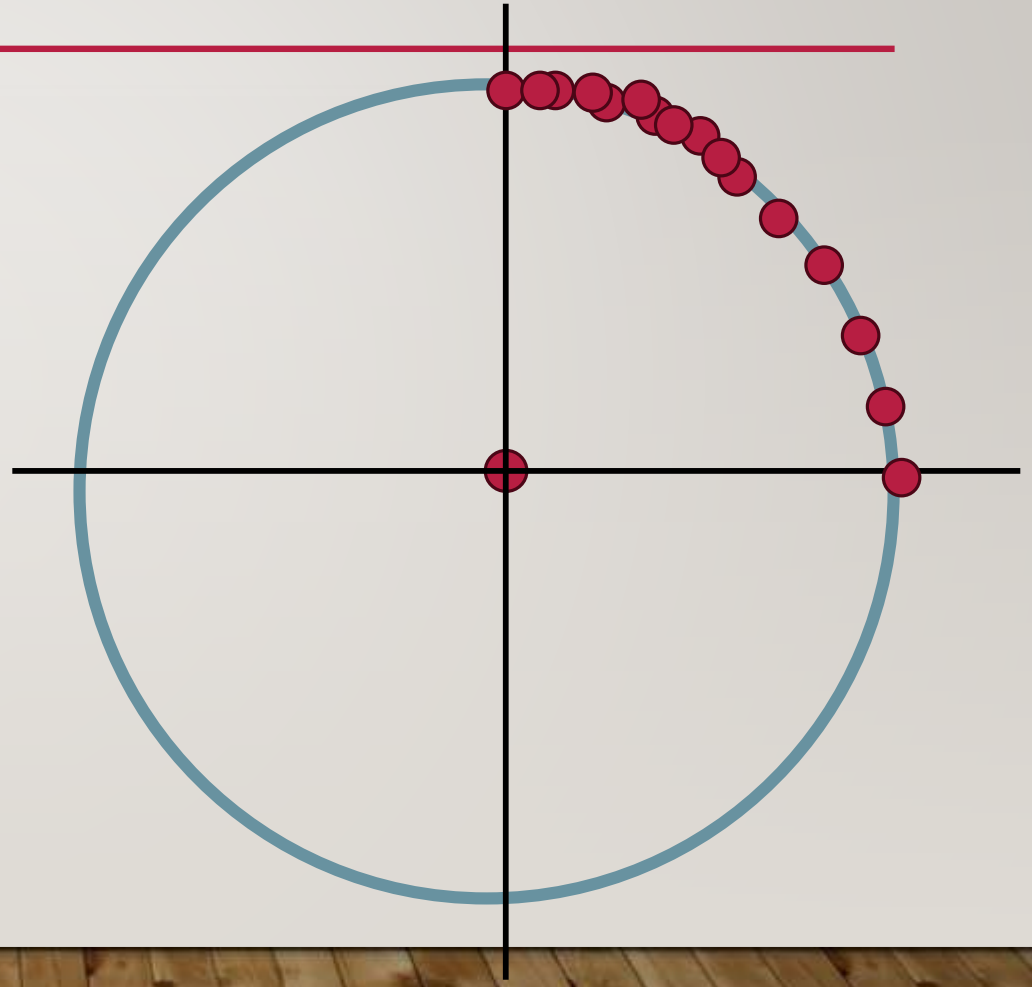
- Find the point with coordinates (px, py) that satisfy the circle by:
- Find all points that have the distance to center equal to (20) by this formula:

  -> 20^2 = px^2 + py^2

  -> py = sqrt(400 – px^2)

  (px from: 0 -> 20)

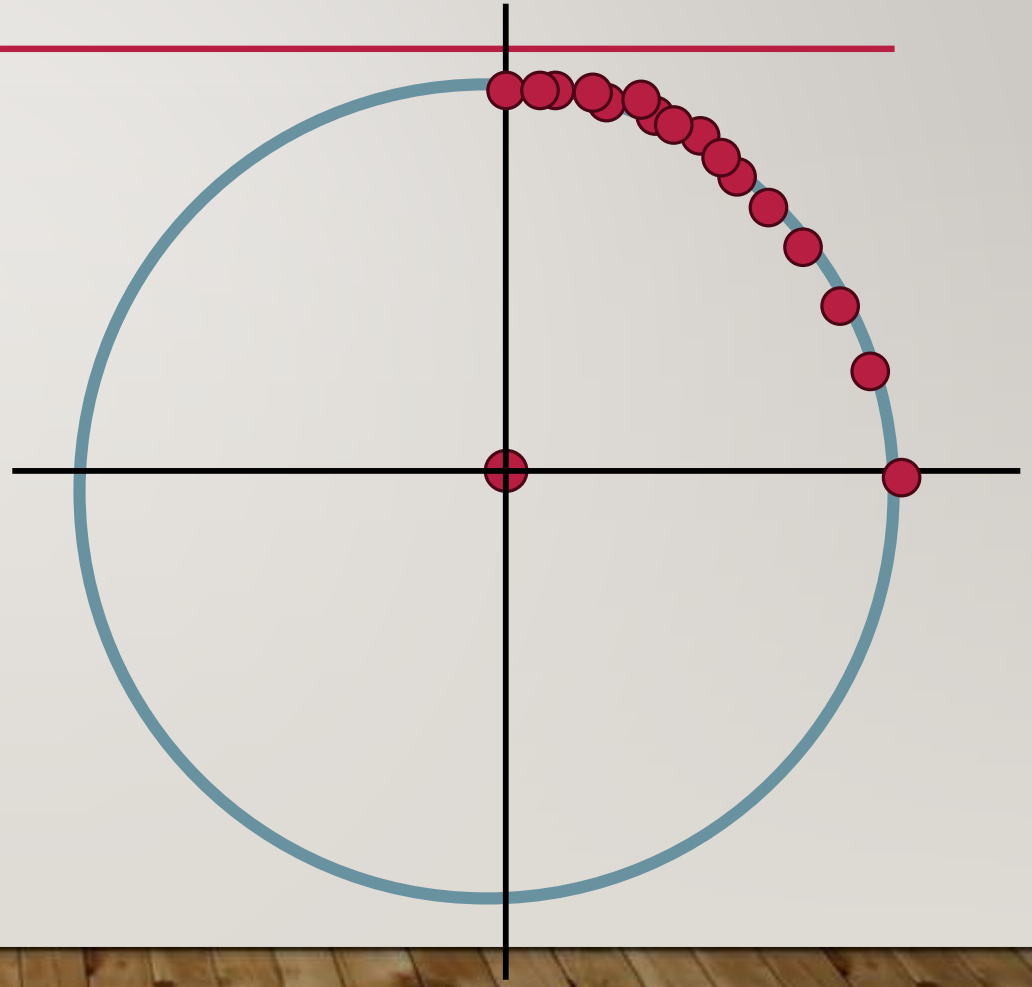
- So we can find all points (px, py) that make up the circle



3) DETAILS OF ALL STEPS
STEP 2: CALCULATE THE RELATIVE POSITIONS OF POINTS TO FORM A CIRCLE

py = sqrt(400 – px^2) (px from: 0 -> 20). So we can calculate all points:

(0, 20)	(1,20)	(2,20)
(3, 20)	(4,20)	(5,19)
(6, 19)	(7,19)	(8,18)
(9, 18)	(10,17)	(11,17)
(12,16)	(13,15)	(14,14)
(15,13)	(16,12)	(17,11)
(18,9)	(19,6)	(20,0)



3) DETAILS OF ALL STEPS
STEP 2: CALCULATE THE RELATIVE POSITIONS OF POINTS TO FORM A CIRCLE

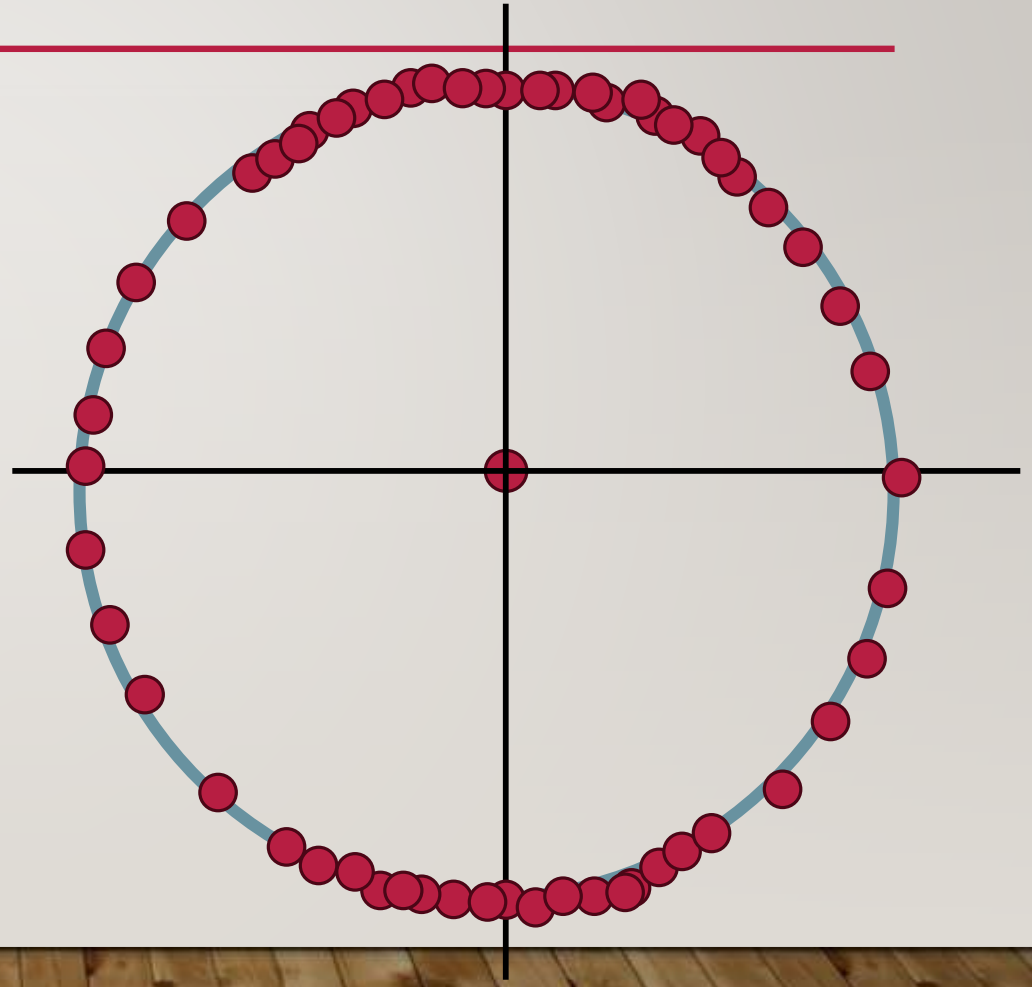
-Because of px and py > 0 (first quadrant) so, let's swap

(-px, py)

(-px, -py)

(px, -py)

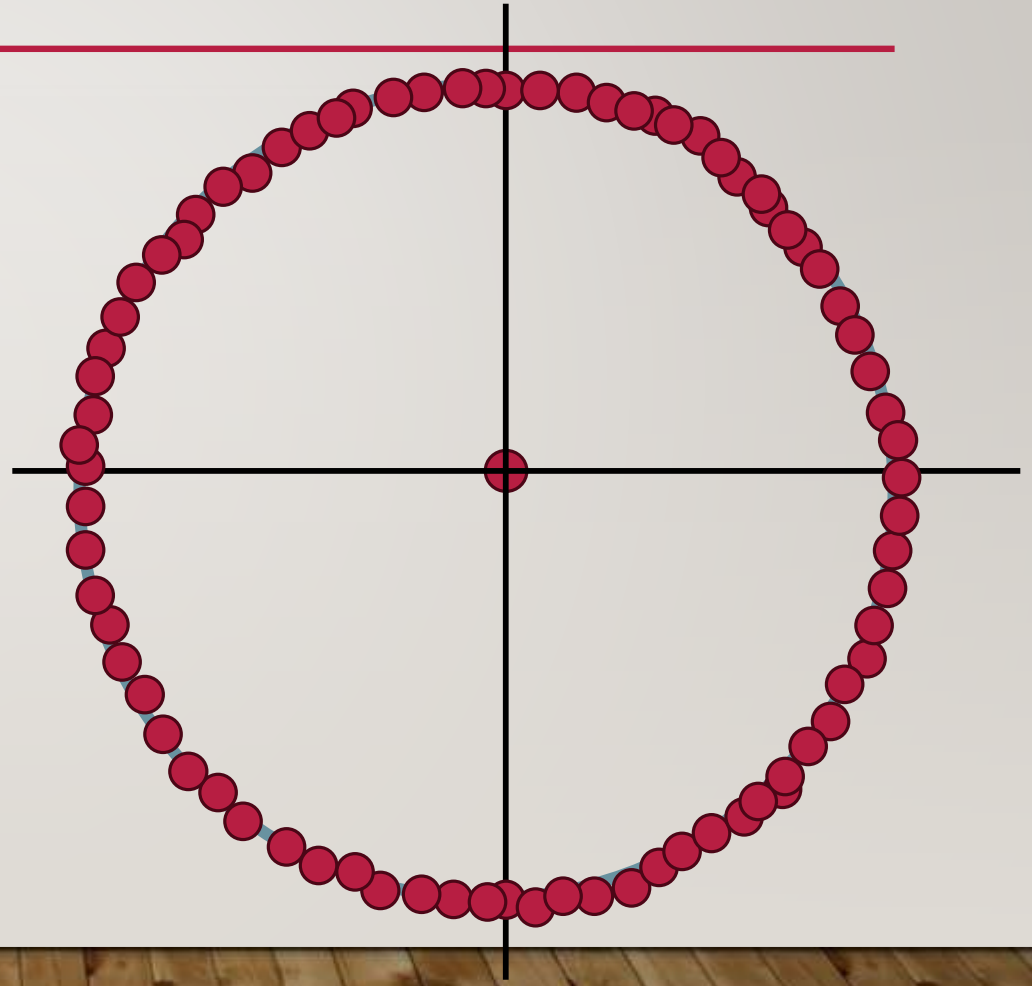
To create all synmetric points



3) DETAILS OF ALL STEPS
STEP 2: CALCULATE THE RELATIVE POSITIONS OF POINTS TO FORM A CIRCLE

-Swap px for py to be symmetrical in the 8th quadrant (Because the algorithm to calculate y earlier was not very precise, so there was no balance between x and y). So once the pair (px, py) is found, 8 other points will be generated. (Circle are full of points now)

-After finding all the points that make up the circle, save all the data to the Circle_points array.



3) DETAILS OF ALL STEPS
STEP 3: CHECK MOVEMENT, INCREASE SPEED:

- - The default of the signal is to keep it in the center of the screen (dx = dy = 0).
- When the move/speed navigation key is received, the value of dx and dy (or sleep time) will be adjusted to prepare for moving from old to new state.
- +) w: up: dx = 0 , dy = -1
- +) s: down: dx = 0, dy = 1
- +) a: left: dx = -1, dy = 0
- +) d: right: dx = 1, dy = 0.
- Speed up: for example, when it's moving up, pressing 'w' will speedup the ball by reducing the sleep time.

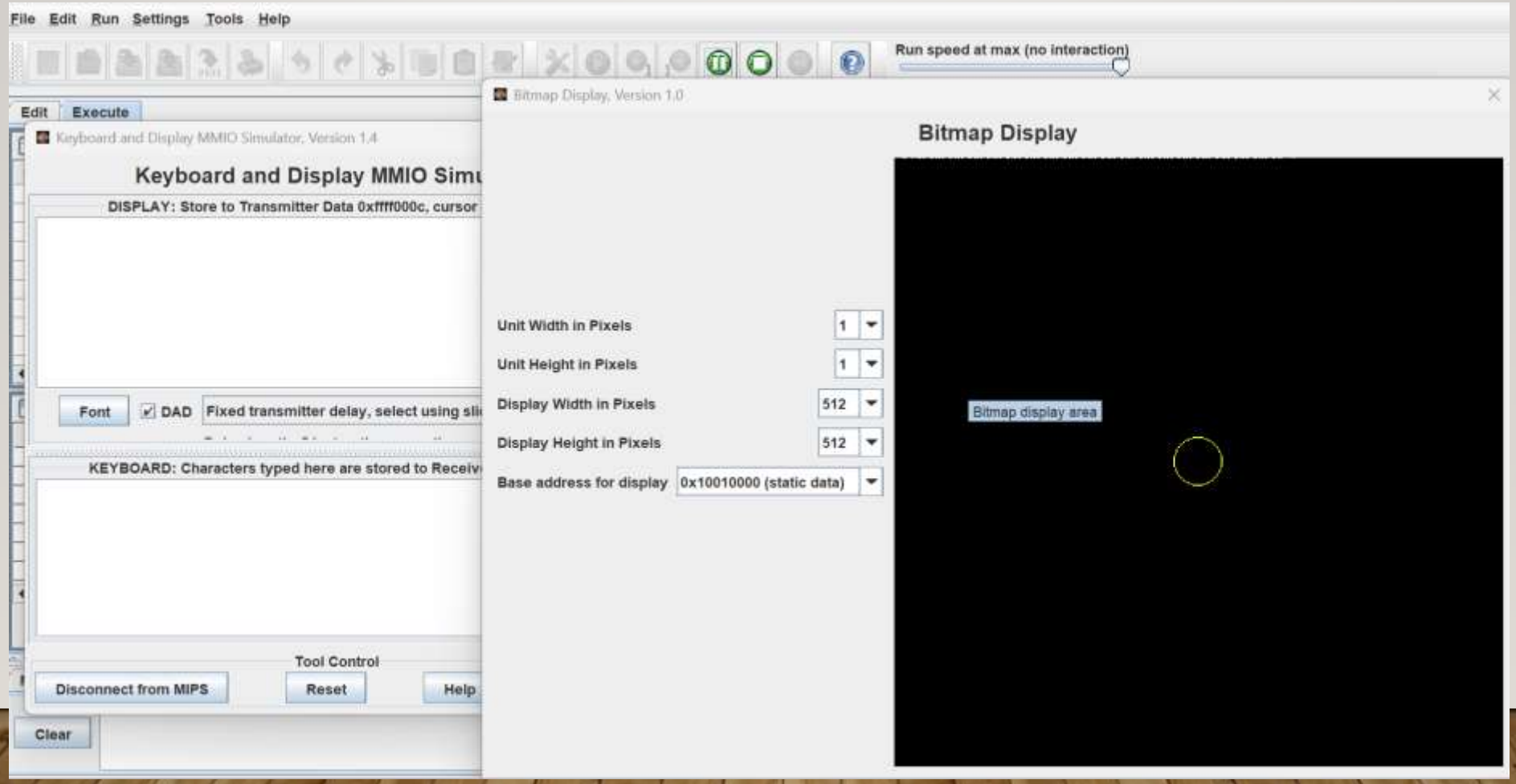
3) OVERVIEW ALL THE STEPS
STEP4: CHECK IF HITS EDGE?

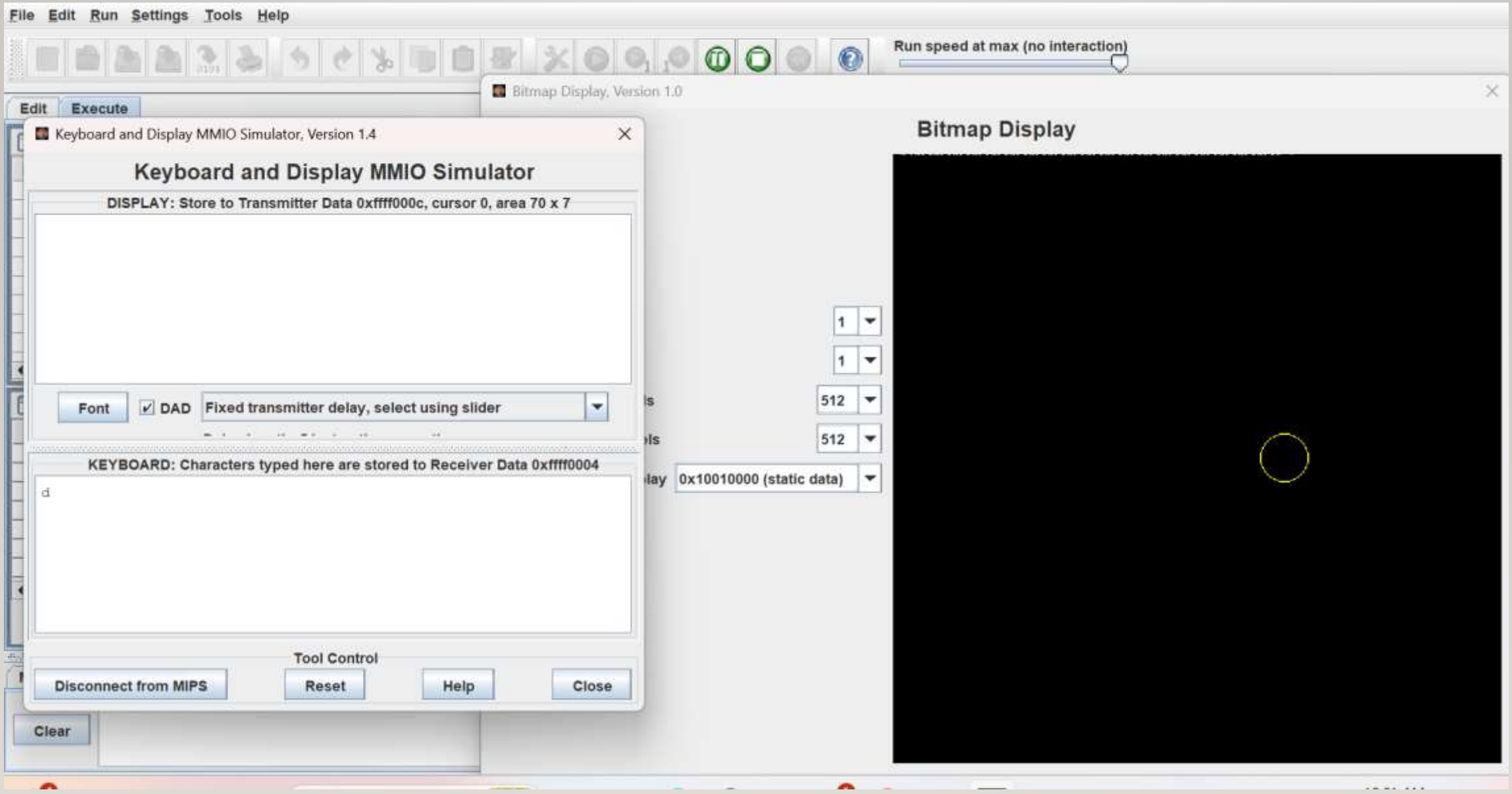
- Check by calculating the center coordinates (x, y) plus the radius that has touched the edge of the screen. If it touches, reverse dx, dy depending on the direction to adjust the movement state.
- For example, in the 4 check functions left right up and down, we check that there is a direction of movement to the right (dx = 1, dy = 0), then we jump to the check function right. Take the center position (x, y) plus the radius R to see if it touches the edge (y + r = 511?) If it does not touch, continue the process, if it does, reset the direction dx dy, then continue.

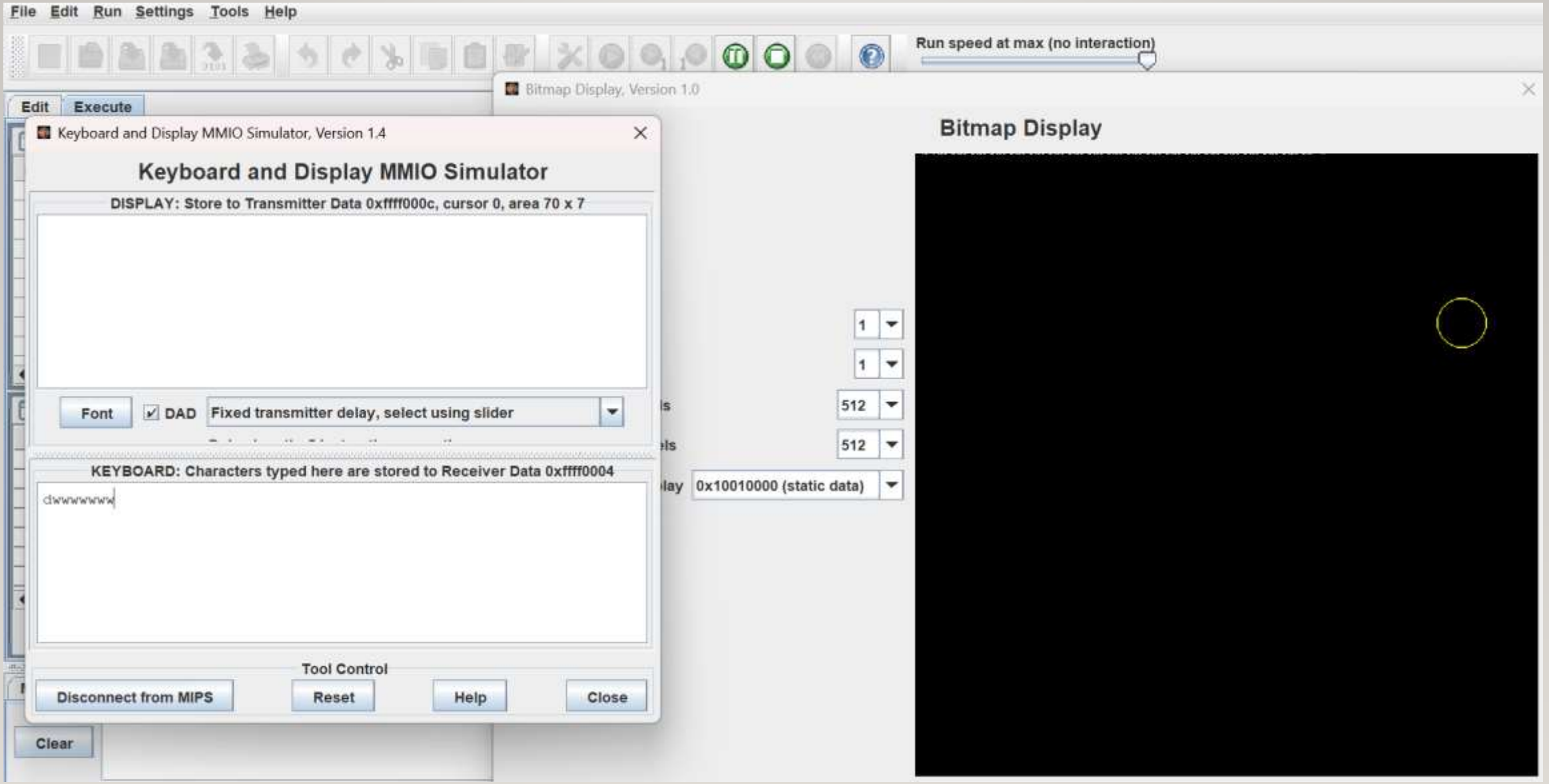
3) OVERVIEW ALL THE STEPS
PHẦN 5: DELETE PREVIOUS CIRCLE, DRAW NEW ONE

- - Delete the circle from the previous by changing the color to black, draw the circle with the existing data (old data), then change color to yellow, update the data and draw again with yellow
- - How to draw a new circle:
- + Calculate the x, y values after adding (dx, dy) in the appropriate direction.
- + Add (px, py) from the circle array to get the points that make up the circle around the center (x, y)
- + Determine the position on the bitmap and proceed to draw.
- + The drawing process stops when the circle_points array reaches the last element (ie all the points have been drawn)

DEMO THE PROJECT




File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Bitmap Display, Version 1.0
Bitmap Display
Keyboard and Display MMIO Simulator, Version 1.4
Keyboard and Display MMIO Simulator
DISPLAY: Store to Transmitter Data 0xffff000c, cursor 0, area 70 x 7
Font
DAD
Fixed transmitter delay, select using slider
KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004
d
Tool Control
Disconnect from MIPS
Reset
Help
Close
Clear
1
1
512
512
0x10010000 (static data)


File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Bitmap Display, Version 1.0
Bitmap Display
Keyboard and Display MMIO Simulator, Version 1.4
Keyboard and Display MMIO Simulator
DISPLAY: Store to Transmitter Data 0xffff000c, cursor 0, area 70 x 7
Font
DAD
Fixed transmitter delay, select using slider
KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004
dwwwwwww
Tool Control
Disconnect from MIPS
Reset
Help
Close
Clear
1
1
512
512
0x10010000 (static data)