Course CS50
Assignment 0 – Pong Game
Mohammad Ali Yaghan -

The first step was to allow the use to select whether he/she wants an automatic paddle, which one, or both (at the start state).

A global variable AUTO_MODE was created to store the user's choice (in the main.lua file) it default value is 0 (i.e. no auto paddels)

```lua
PADDLE_SPEED = 200

-- Auto mode
AUTO_MODE = 0 --the auto mode tells the game if the paddels are automaticaly played in what sense
              -- 0 -> no paddel is auto
              -- 1 -> left paddel is auto
              -- 2 -> right paddel is auto
              -- 3 -> both paddels are auto
--[[
    Called just once at the beginning of the game; used to set up
    game objects, variables, etc. and prepare the game world.
]]
function love.load()
    -- set love's default filter to "nearest-neighbor", which essentially
```

A call to a new function called: "determineAuto" from within "love.keypressed" was placed

```lua
    ]]--
function love.keypressed(key)
    -- `key` will be whatever key this callback detected as pressed
    if key == 'escape' then
        -- the function LÖVE2D uses to quit the application
        love.event.quit()
    -- if we press enter during either the start or serve phase, it should
    -- transition to the next appropriate state
    elseif key == 'enter' or key == 'return' then
        if gameState == 'start' then
            gameState = 'serve'
        elseif gameState == 'serve' then
            gameState = 'play'
        elseif gameState == 'done' then
            -- game is simply in a restart phase here, but will set the serving
            -- player to the opponent of whomever won for fairness!
            gameState = 'serve'
            ball:reset()
            -- reset scores to 0
            player1Score = 0
            player2Score = 0
            -- decide serving player as the opposite of who won
            if winningPlayer == 1 then
                servingPlayer = 2
            else
                servingPlayer = 1
            end
        end
    else
        determineAuto(key)
    end

end

function determineAuto(key)
    if (key == 'L' or key =='l') and (gameState == 'start' or gameState == 'done') then
        AUTO_MODE = 1
    elseif (key == 'R' or key =='r') and (gameState == 'start' or gameState == 'done') then
        AUTO_MODE = 2
    elseif (key == 'B' or key =='b') and (gameState == 'start' or gameState == 'done') then
        AUTO_MODE = 3
    end

end
```

and a line was added to “love.draw” function to aske for user’s input

```lua
    drawing all of our game objects and more to the screen.
]]
function love.draw()
    -- begin drawing with push, in our virtual resolution
    push:start()

    -------love.graphics.setBackgroundColor(100, 45, 55, 255)
    love.graphics.clear(0, 0, 0, 255)

    -- render different things depending on which part of the game we're in
    if gameState == 'start' then
        -- UI messages
        love.graphics.setColor(0, 255, 0, 255)
        love.graphics.setFont(smallFont)
        love.graphics.printf('Welcome to Pong!', 0, 10, VIRTUAL_WIDTH, 'center')
        love.graphics.printf('Press Enter to begin!', 0, 20, VIRTUAL_WIDTH, 'center')
        love.graphics.printf('Press R or L to pick the Auto Paddel - B for both !', 0, 30, VIRTUAL_WIDTH, 'center')
    elseif gameState == 'serve' then
        -- UI messages
        love.graphics.setFont(smallFont)
        love.graphics.printf('Player ' .. tostring(servingPlayer) .. "'s serve!",
        0, 10, VIRTUAL_WIDTH, 'center')
```

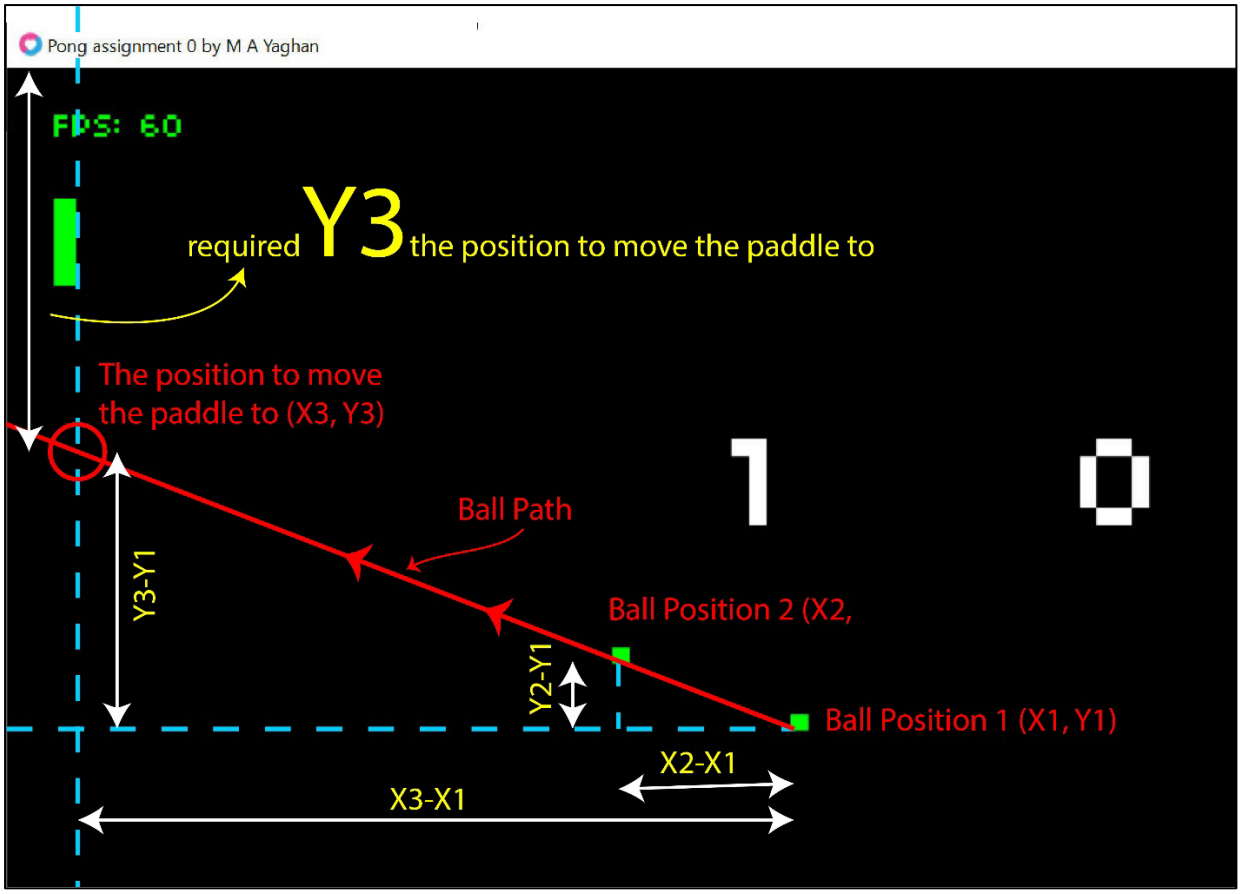
In the “love.update” where the paddles are moved, a condition to check the AUTO_MODE value was added to decide whether to allow any paddle to be moved by the key’s input

main.lua | Paddle.lua | Ball.lua

```lua
    end

    --
    -- paddles can move no matter what state we're in
    --
    -- player 1
    if AUTO_MODE == 0 or AUTO_MODE == 2 then --player1 is not auto
        if love.keyboard.isDown('w') then
            player1.dy = -PADDLE_SPEED
        elseif love.keyboard.isDown('s') then
            player1.dy = PADDLE_SPEED
        else
            player1.dy = 0
        end
    else -- player 1 is in auto mode
        if ball.x - ball.PrevX < 0 then --the ball is moving towards player1
            player1:movePaddleTo (findBallintersecting (ball.x, ball.y, ball.PrevX, ball.PrevY, 10))
        elseif ball.x - ball.PrevX > 0 then --the ball is moving away
            player1:movePaddleTo(VIRTUAL_HEIGHT / 2)
        -- if it was 0 then do nothing it is the initial stage
        end
    end

    -- player 2
    if AUTO_MODE == 0 or AUTO_MODE == 1 then --palyer2 is not auto
        if love.keyboard.isDown('up') then
            player2.dy = -PADDLE_SPEED
        elseif love.keyboard.isDown('down') then
            player2.dy = PADDLE_SPEED
        else
            player2.dy = 0
        end
    else -- player2 is in auto mode
        if ball.x - ball.PrevX > 0 then --the ball is moving towards player2
            player2:movePaddleTo(findBallintersecting (ball.x, ball.y, ball.PrevX, ball.PrevY, VIRTUAL_WIDTH-10))
        elseif ball.x - ball.PrevX < 0 then --ball is moving away form player2
            player2:movePaddleTo(VIRTUAL_HEIGHT / 2)
        -- if it was 0 then do nothing it is the initial stage
        end
    end

    -- update our ball based on its DX and DY only if we're in play state;
    -- scale the velocity by dt so movement is framerate-independent
    if gameState == 'play' then
        ball:update(dt)
```

length : 15,261 lines : 466 Ln : 351 Col : 25 Sel : 15 | 1 Unix (

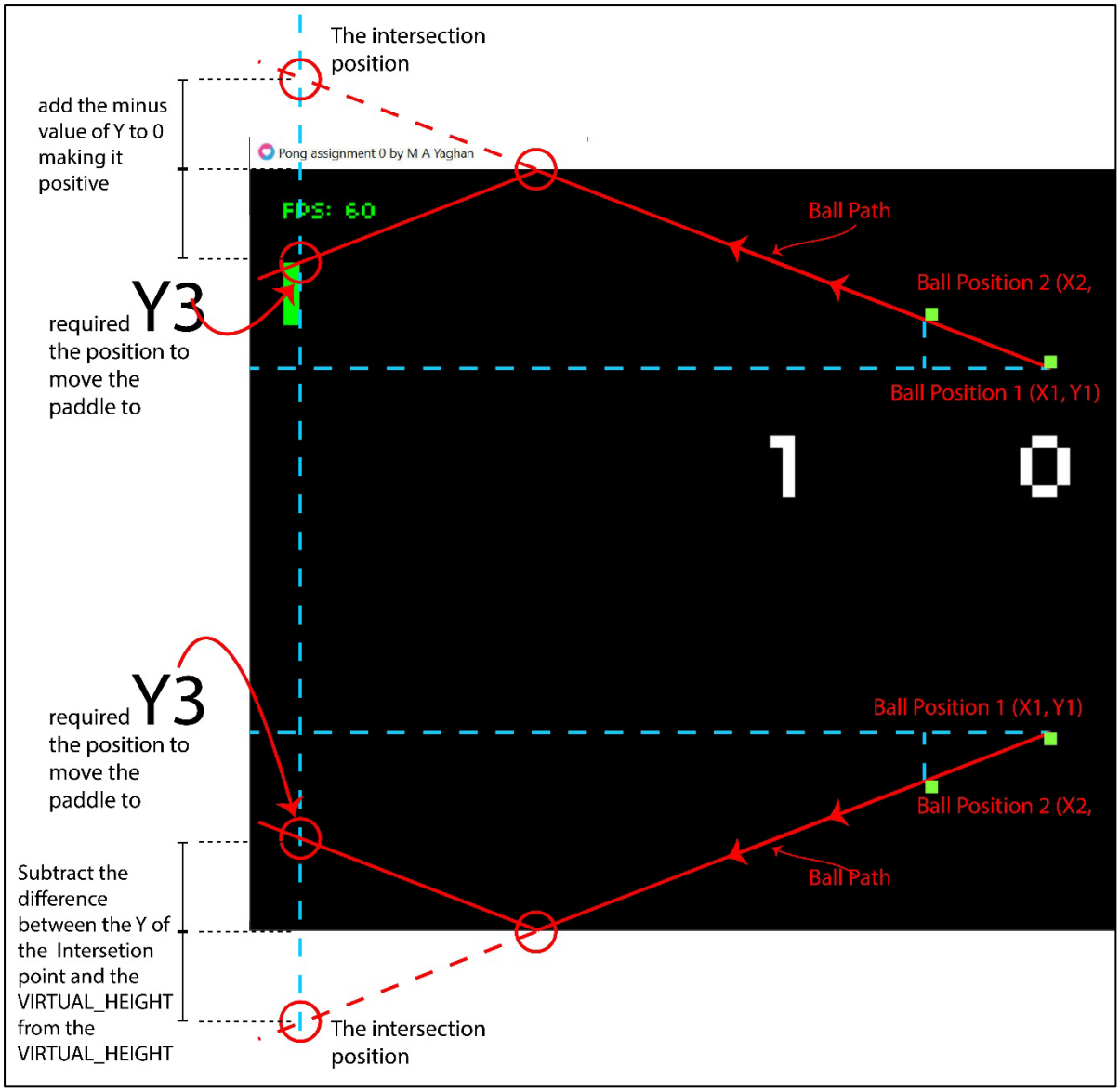
Second, the logic to anticipate where to move the paddle to be able to collide with the ball can be explained in the following drawing



According to two positions of the ball, its straight path will collide with the vertical line of the paddle (here the left paddle, but the logic works for both). The Y value for the intersection point (Y3 in the figure) will be deducted from the following formula

$$Y3 = \frac{(Y2 - Y1)}{(X2 - X1)} \times (X3 - X1) + Y1$$

In case of the intersection point was bellow zero or higher than the VIRTUAL_HEIGHT then a reflection in the path occurs according to the following figure



If Y3 < 0 then Y3 should equal abs(Y3)

If Y3 > VIRTUAL_HEIGHT then

Y3 should equal VIRTUAL_HEIGHT - (Y3- VIRTUAL_HEIGHT)

In order to determine the ball path two extra attributes has to be added to the ball class (PrevX and PrevY) in the initialization of Ball:init

```lua
Ball = Class{}

function Ball:init(x, y, width, height)
    self.x = x
    self.y = y
    self.width = width
    self.height = height
    self.PrevX = self.x
    self.PrevY = self.y

    -- these variables are for keeping track
    -- X and Y axis, since the ball can move
    self.dy = 0
    self.dx = 0
end
```

And they will be updated in the “Ball:update” function

```lua
end

function Ball:update(dt)
    self.PrevX = self.x
    self.PrevY = self.y
    self.x = self.x + self.dx * dt
    self.y = self.y + self.dy * dt
end
```

To find the position to move the paddle to, I added a function in the main file

```lua
function findBallintersecting (bx, by, bxpast, bypast, Xedge)

    if bxpast == bx then

        return VIRTUAL_HEIGHT / 2
    else
        Yinter = ((Xedge - bx)*(bypast - by))/(bxpast - bx) + by

        if (Yinter > 0) and (Yinter < VIRTUAL_HEIGHT) then -- it is within the wall
            return Yinter
        elseif Yinter < 0 then
            Yinter = -Yinter -- it should reflect
            return Yinter
        else
            Yinter = VIRTUAL_HEIGHT - (Yinter - VIRTUAL_HEIGHT) -- it should rflect
            return Yinter
        end
    end
    return VIRTUAL_HEIGHT / 2 --in case of an unexpected situation retun the mid poi
end
```

It will return the position as explained or the mid-point is case of undefined situations.

Then we will need a function that actually moves the paddle to a predefined position. I added Paddle:movePaddleTo (YPos) to the Paddle class

```lua

-- a function that moves the paddle towards a defined YPos
function Paddle:movePaddleTo (YPos)
    Mvalue = math.random(1, AUTO_MISTAKE) -- the value to make a mistake
    if self.y > YPos + Mvalue then
        self.dy = -PADDLE_SPEED
    elseif self.y < YPos - Mvalue then
        self.dy = PADDLE_SPEED
    else -- the paddle is in the YPos
        self.dy = 0 -- here try to make tollerence
    end
end


```

AUTO_MISTAKE is a global variable defined in the main fine that is randomized every time making it possible for the paddle to miss on the ball.

```lua
VIRTUAL_HEIGHT = 244

AUTO_MISTAKE = 120 -- a random number to create a mistake in the auto play mode


```

The greater the value the less accurate the paddle is.

Finally, you can play the game, the files are attached, THANK YOU.

Mohammad Ali Yaghan

3rd March 2020