================BAYRAM'S PART================

* Creating, designing and editing the visuals of the game. such as menus or objects

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_PUZZLE GAME\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* We wanted to design a puzzle game. We cut 4 shapes in 4 different colors for one of our pictures. We wrote that shape and color in each picture space we cut.
* The game consists of 4 objects and 1 background. You win when you bring the objects to the right place and ensure the integrity of the picture. You use the mouse to move these objects.

# Square - 220 - Green

# Rectangle - 300 100 - Yellow

# Circle – 110 - Red

# Triangle - 220 - Blue

\_\_\_\_\_\_\_\_\_\_ MOUSE + GAME-SCENE (AS A PROPORTY) + DRAGGİNG\_\_\_\_\_\_\_\_\_

; purpose: check whether the point on the button or not

;contract: isInButton: b(button) ) x(number) y(number) -> boolean

isInButton🡪 control mouse tip point

; purpose: check whether mouse clicked button or not

; contract: doesClickButton: b(button) x(number) y(number) mo(mouseEvent)-> boolean

doesClickButton🡪To check whether the mouse clicked and grabbed the object

; purpose: check whether the point on the rectangle or not

; contract: isInRectangle: rec(rectangleOb) x(number) y(number) -> boolean

isInRectangle🡪 To check whether the mouse tip is on its rectangle or not

; purpose: check whether the point on the circle or not

; contract: isInCircle: circ(circleOb) x(number) y(number) -> boolean

isInCircle🡪 To check whether the mouse tip is on its circle or not

; purpose: check whether the point on the triangle or not

; contract: isInTriangle: tri(triangleOb) x(number) y(number) -> boolean

isInTriangle🡪 To check whether the mouse tip is on its triangle or not

; purpose: check whether the point on the square or not

; contract: isInTriangle: sq(squareOb) x(number) y(number) -> boolean

isInSquare🡪 To check whether the mouse tip is on its square or not

; purpose: check whether mouse is dragged or not

; contract: doesDrag: pg(puzzleGame) x(number) y(number) mo(mouseEvent)-> boolean

doesDrag🡪 We wrote a mouse move function to hold and drag objects with the mouse and drop them into the correct space.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_PUZZLE REACTİONS\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; purpose: calculating the distance between the positions

; contract: distCalc: pos1(position) pos2(position) -> number

distCalc🡪 We wrote this code to measure the distance between the initial location and the target location.

; purpose: determinating whether puzzle are completed or not

; contract: doesFillUp: sq(squareOb) triang(trinagleOb) rect(rectangleOb) circ(circleOb) -> boolean

doesFillUp🡪 To check whether the puzzle has all the objects in the correct position. To understand whether a picture has been created

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ PUZZLE GAME BACKGROUND DESIGNING\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; purpose: drawing the puzzle game

; contract: finalScenePuzzle: ga(puzzleGame) -> image

finalScenePuzzle🡪to create the puzzle game with a background and draw the game