**Game Plan**

Ray Bike Supreme

**Game Description**

The game starts with a main landing page where the players can find the game play instructions, enter their name and select the game difficulty before pressing the start game button. When the game starts, the two players will control their bikes using the keyboard buttons (asdw for player one and up-down-left-right for player two). The goal of the game is to use the trail that each of the bikes leaves behind to trap the other player and force that player to run into it. The score earned by the winner will be the number of seconds the game lasted. When the game is over, the players can see their highest score so far and their accumulated score from the game over page. From that page, the players can choose to play again or return to the main menu.

**Screen by Screen Description of the UI**

A screenshot of a computer screen

Description automatically generated

HTML Elements:

title h1 element

help button element

“Player” p element

name input elements

bike image elements

“Difficulty” p element

Difficulty selection Nav bar element

Start game button element

A screenshot of a computer instruction screen

Description automatically generated

HTML Elements:

Modal div element

Close screen button element

Instruction p elements

A screenshot of a video game

Description automatically generated

HTML Elements:

“Player” text p elements

Player name p elements

Score p elements

Arena div elements

Bike image elements

Rays canvas elements

Rock image elements

A screenshot of a computer game

Description automatically generated

HTML Elements:

Modal elements

Text and score stats p elements

Play again button element

Return to Menu button elements

**Breakdown of Functional Components**

Main screen

- Player name input section

- difficulty selection menu

- Help/Instruction button

- Start game button

Instruction screen

- game play instruction text

- exist screen button

Game screen

- Scoreboard

- Gameplay arena

Game over screen

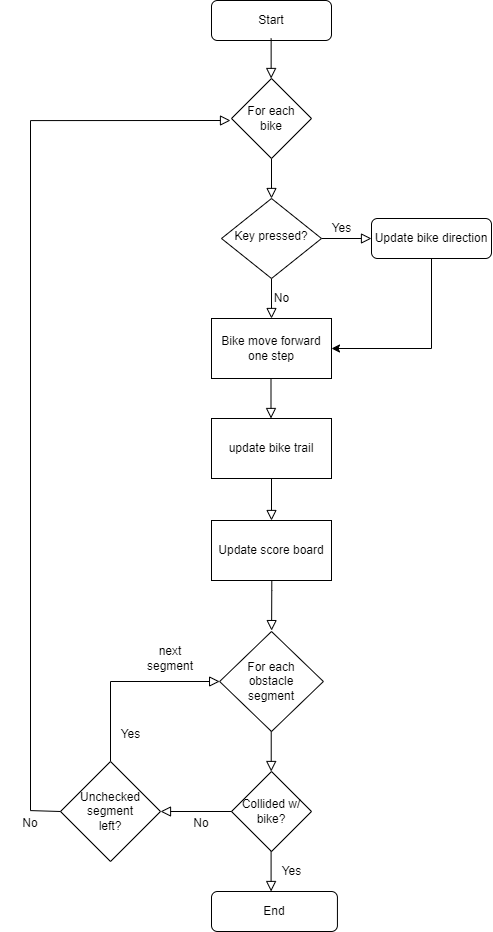
- current game score

- accumulated game score

- return to main screen button (erases accumulated score)

- play again button

**Gameplay logic flow chart:**



**Glossary of Class, Constants, Variables and Methods:**

class Game {

//constants

\_SEGLENGTH :number //intrinsic segment length of the game

\_RAYWIDTH :number //width of bike ray

\_BIKESPEED :number //number of pixel advance per game logic loop

\_ARENA\_WIDTH :number //pixel width of gameplay arena

\_ARENA\_HEIGHT :number //pixel height of gameplay arena

\_ARENA\_CEN\_POS :number //[left, top] of arena’s center

\_GAME\_START\_TIME :Date //Datetime object of the game start time

\_MIN\_OBS\_HEIGHT :number //min pixel height of rocks in medium difficulty mode

\_MAX\_OBS\_HEIGHT :number //max pixel height of rock in medium difficulty mode

\_OBS\_IMG\_PATH :string //path of the rock image

\_BIKE\_IMG\_PATH :string //path of bike image

//variables

\_difficulty :number //1 to 3 for easy to hard game mode, respectively

**\_**score:number // track score of the game

**\_**obsSegments :number [][] //array of obstacles segments, each segment is an array [x1,

y1, x2, y2]

\_trailCanvases: HTMLElement //list of canvases html elements used to draw bike trails

\_arena: HTMLElement //html element of the gameplay arena

\_bikes: Bike[] //list of Bike objects

//methods

\_setupArena() //create div element with appropriate css tags then add to DOM

\_setupScoreBoard() //create div element with appropriate css tags then add to DOM

\_setupCanvases() //create canvases elements in DOM and also add them to

\_trailCanvases variable

\_createBikes() //add bike images DOM with appropriate position and create

Bike object and added to \_bikes list

\_addObstacles() //randomly places obstacles (rock) on arena by appending rock

images element to arena elements, also add image

boundaries to \_obsSegment array

\_checkImgOverlap() //return Boolean of weather two obstacles(rock) image overlaps

\_evolveGame() //initiate game logic loop with a indefinitely while loop

\_incrementScore() //increment score board score

\_draw\_Trail() //draw or redraw bike trails

}

Class Bike {

//constants

\_DIR\_ARRAY :string[] //array of directions ordered by how it evolves as as user hits the

right key

\_BikeRotation :enum //enum relating bike direction and degrees of image rotation

//variables

\_arena :HTMLElement //arena html element

\_bikeElement :HTMLELEMENT //img html element of the bike

\_imgPosition :number[] //[left,top] of img when it's first loaded

\_imgWidth :number //img width when it's first loaded

\_imgHeight :number //img height when it's first loaded

\_kbControl :string[] //an array [up,down,left,right] keyboard control key of bike

\_headPosition :number[] //[x,y] position of bike's head

\_centerPosition :number[] //[x,y] position of bike's center

\_tailPosition :number[] //[x,y] position of bike's tail

\_direction :string //current direction of bike's motion

\_speed :number //num pixel bike moves per game interation

\_bikeId :string //id field of bike's img html element

\_centerSeg :number[] //[x\_old, y\_old, x\_new, y\_new],evolution of bike center position

during last interation

\_trail :number[][] // a list of [x1,y1,x2,y2] segments the bike has travelled over

\_trailColor :string //color of the trail

\_cttSegNum :number //number of segments needed to span from bike center to bike tail

//methods

\_getHeadPosition() :number[] //get list [x,y] of bike’s head position

\_getCenterPostion() :number[] //get list [x,y] of bike’s center position

\_getTailPosition() :number[] //get list [x,y] of bike’s tail position

\_calculateHeadPosition() :number[] //get list [x,y] of head position base on current image

placement

\_calculateCenterPosition() :number[] //get list [x,y] of center position base on current image

placement

\_calculateTailPosition() :number[] //get list [x,y] of tail position base on current image

placement

\_getImgPosition() :number[] //get list [left,top] of the bike image on the page

\_updateDirection() //update bike direction and image placement base on key

pressed

\_getNewDirection() :str //determine new direction using DIR\_ARRAY

getTrail() :number[][] //get list of [x1,y1,x2,y2] of trail segments left behind by

the bike

getTrailColor() :str //getter method for trail color

moveForward() //advance bike’s motion along its current direction

hasCollided(obsSegment:number[][]):bool //determine if a bike has collided with the list of

obstacles segments

}