

# SE 3XA3: Module Interface Specification

## Mini-Arcade

Andrew Hum	William Lei	Arshan Khan	Jame Tran
huma3	leim5	khana172	tranj52
400138826	400125240	400145605	400144141

# Contents

<b>1</b>	<b>MIS of Launcher Module</b>	<b>5</b>
1.1	Interface Syntax . . . . .	5
1.1.1	Exported Access Programs . . . . .	5
1.2	Interface Semantics . . . . .	5
1.2.1	State Variables . . . . .	5
1.2.2	Assumptions . . . . .	5
1.2.3	Access Program Semantics . . . . .	6
<b>2</b>	<b>MIS of Scoreboard Module</b>	<b>6</b>
2.1	Interface Syntax . . . . .	6
2.1.1	Exported Access Programs . . . . .	6
2.2	Interface Semantics . . . . .	6
2.2.1	State Variables . . . . .	6
2.2.2	Assumptions . . . . .	7
2.2.3	Access Program Semantics . . . . .	7
<b>3</b>	<b>MIS of Maze Generation Module</b>	<b>8</b>
3.1	Interface Syntax . . . . .	8
3.1.1	Exported Access Programs . . . . .	8
3.2	Interface Semantics . . . . .	8
3.2.1	State Variables . . . . .	8
3.2.2	Environment Variables . . . . .	8
3.2.3	Assumptions . . . . .	8
3.2.4	Access Program Semantics . . . . .	9
<b>4</b>	<b>MIS of Score Tracking (Maze) Module</b>	<b>9</b>
4.1	Interface Syntax . . . . .	9
4.1.1	Exported Access Programs . . . . .	9
4.2	Interface Semantics . . . . .	9
4.2.1	State Variables . . . . .	9
4.2.2	Environment Variables . . . . .	10
4.2.3	Assumptions . . . . .	10
4.2.4	Access Program Semantics . . . . .	10
<b>5</b>	<b>MIS of Draw Game (Maze) Module</b>	<b>10</b>
5.1	Interface Syntax . . . . .	10
5.1.1	Exported Access Programs . . . . .	10
5.2	Interface Semantics . . . . .	10
5.2.1	State Variables . . . . .	10
5.2.2	Environment Variables . . . . .	10
5.2.3	Assumptions . . . . .	11

5.2.4	Access Program Semantics . . . . .	11
<b>6</b>	<b>MIS of Player Movement (Maze) Module</b>	<b>11</b>
6.1	Interface Syntax . . . . .	11
6.1.1	Exported Access Programs . . . . .	11
6.2	Interface Semantics . . . . .	11
6.2.1	State Variables . . . . .	11
6.2.2	Environment Variables . . . . .	12
6.2.3	Assumptions . . . . .	12
6.2.4	Access Program Semantics . . . . .	12
<b>7</b>	<b>MIS of Menu and Settings (Maze) Module</b>	<b>13</b>
7.1	Interface Syntax . . . . .	13
7.1.1	Exported Access Programs . . . . .	13
7.2	Interface Semantics . . . . .	13
7.2.1	State Variables . . . . .	13
7.2.2	Environment Variables . . . . .	13
7.2.3	Assumptions . . . . .	13
7.2.4	Access Program Semantics . . . . .	13
<b>8</b>	<b>MIS of Ball Tracking Module</b>	<b>14</b>
8.1	Interface Syntax . . . . .	14
8.1.1	Exported Access Programs . . . . .	14
8.2	Interface Semantics . . . . .	14
8.2.1	State Variables . . . . .	14
8.2.2	Assumptions . . . . .	14
8.2.3	Access Program Semantics . . . . .	14
<b>9</b>	<b>MIS of Score Tracking (Pong) Module</b>	<b>15</b>
9.1	Interface Syntax . . . . .	15
9.1.1	Exported Access Programs . . . . .	15
9.2	Interface Semantics . . . . .	15
9.2.1	State Variables . . . . .	15
9.2.2	Assumptions . . . . .	15
9.2.3	Access Program Semantics . . . . .	15
<b>10</b>	<b>MIS of Draw Game (Pong) Module</b>	<b>16</b>
10.1	Interface Syntax . . . . .	16
10.1.1	Exported Access Programs . . . . .	16
10.2	Interface Semantics . . . . .	16
10.2.1	State Variables . . . . .	16
10.2.2	Access Program Semantics . . . . .	16

<b>11 MIS of Player Movement (Pong) Module</b>	<b>17</b>
11.1 Interface Syntax . . . . .	17
11.1.1 Exported Access Programs . . . . .	17
11.2 Interface Semantics . . . . .	17
11.2.1 State Variables . . . . .	17
11.2.2 Environment Variables . . . . .	17
11.2.3 Assumptions . . . . .	17
11.2.4 Access Program Semantics . . . . .	17
<b>12 MIS of Menu and Settings (Pong) Module</b>	<b>18</b>
12.1 Interface Syntax . . . . .	18
12.1.1 Exported Access Programs . . . . .	18
12.2 Interface Semantics . . . . .	18
12.2.1 State Variables . . . . .	18
12.2.2 Environment Variables . . . . .	18
12.2.3 Assumptions . . . . .	18
12.2.4 Access Program Semantics . . . . .	19
<b>13 MIS of Score Tracking (Flappy) Module</b>	<b>19</b>
13.1 Interface Syntax . . . . .	19
13.1.1 Exported Access Programs . . . . .	19
13.2 Interface Semantics . . . . .	19
13.2.1 State Variables . . . . .	19
13.2.2 Assumptions . . . . .	19
13.2.3 Access Program Semantics . . . . .	19
<b>14 MIS of Map Generation (Flappy) Module</b>	<b>20</b>
14.1 Interface Syntax . . . . .	20
14.1.1 Exported Access Programs . . . . .	20
14.2 Interface Semantics . . . . .	20
14.2.1 State Variables . . . . .	20
14.2.2 Environment Variables . . . . .	20
14.2.3 Assumptions . . . . .	20
14.2.4 Access Program Semantics . . . . .	20
<b>15 MIS of Draw Game (Flappy) Module</b>	<b>21</b>
15.1 Interface Syntax . . . . .	21
15.1.1 Exported Access Programs . . . . .	21
15.2 Interface Semantics . . . . .	21
15.2.1 State Variables . . . . .	21
15.2.2 Environment Variables . . . . .	21
15.2.3 Assumptions . . . . .	21
15.2.4 Access Program Semantics . . . . .	21

<b>16 MIS of Player Movement (Flappy) Module</b>	<b>22</b>
16.1 Interface Syntax . . . . .	22
16.1.1 Exported Access Programs . . . . .	22
16.2 Interface Semantics . . . . .	22
16.2.1 State Variables . . . . .	22
16.2.2 Environment Variables . . . . .	22
16.2.3 Assumptions . . . . .	22
16.2.4 Access Program Semantics . . . . .	22
<b>17 MIS of Menu and Settings (Flappy) Module</b>	<b>23</b>
17.1 Interface Syntax . . . . .	23
17.1.1 Exported Access Programs . . . . .	23
17.2 Interface Semantics . . . . .	23
17.2.1 State Variables . . . . .	23
17.2.2 Environment Variables . . . . .	23
17.2.3 Assumptions . . . . .	23
17.2.4 Access Program Semantics . . . . .	24

Table 1: **Revision History**

Date	Version	Notes
3/9/2020	1.0	Arshan and Andrew created document and sections
3/11/2020	1.1	Andrew completed all modules relevant to Maze
3/12/2020	1.2	Arshan updated all modules relevant to Pong
3/12/2020	1.3	William completed modules relevant to launcher and scoreboard.
3/13/2020	1.4	Jame completed modules relevant to Flappy

Level 1	Level 2
Hardware-Hiding Module	
Behaviour-Hiding Module	Launcher Modules Scoreboard Modules Draw Game (Maze) Player Movement (Maze) Menu and Settings (Maze) Draw Game (Pong) Player Movement (Pong) Menu and Settings (Pong) Draw Game (Flappy) Player Movement (Flappy) Menu and Settings (Flappy)
Software Decision-Hiding Module	Maze Generator (Maze) Score Tracking (Maze) Ball Trajectory (Pong) Score Tracking (Pong) Map Generator (Flappy) Score Tracking (Flappy)

Table 2: Module Hierarchy

# 1 MIS of Launcher Module

## 1.1 Interface Syntax

### 1.1.1 Exported Access Programs

Name	In	Out	Exceptions
displayLauncher	-	GUI	-
launchGame	integer	-	-
launchScoreboard	-	-	-

## 1.2 Interface Semantics

### 1.2.1 State Variables

None

### 1.2.2 Assumptions

None

### 1.2.3 Access Program Semantics

displayLauncher():

Input: None

Transition: draws the launcher to the output window and display it.

Output: the launcher's UI to the screen.

Exceptions: None

launchGame(gameID):

Input: integer representing the game to be launched

Transition: call the drawInterface function of the mini-game (depending on the input integer).

Output: None

Exceptions: None

launchScoreboard():

Input: None

Transition: call the drawScoreboard function of the scoreboard.

Output: None

Exceptions: None

## 2 MIS of Scoreboard Module

### 2.1 Interface Syntax

#### 2.1.1 Exported Access Programs

Name	In	Out	Exceptions
drawScoreboard	integer	GUI	-
changeGame	integer	-	-
exitScoreBoard	-	-	-
readData	-	-	FileNotFound, FileCannotRead
writeData	-	FileSystem	FileCannotWrite
updateScore	integer, integer	-	-
highScore	integer	integer	-

### 2.2 Interface Semantics

#### 2.2.1 State Variables

displayGame: integer - representing the game the scoreboard will be displayed for.

exitScreen: integer - which screen will it display when it exits.

scores: array of arrays of integer - each array stores high scores of a game in descending order.

### 2.2.2 Assumptions

The input to each function (integers) are in appropriate range.

### 2.2.3 Access Program Semantics

drawScoreboard(screenID):

Input: integer representing which screen was displayed before scoreboard.

Transition: set the value of exitScreen to screenID, and draws the scoreboard to the output window and display it,

Output: the scoreboard's UI to the screen.

Exceptions: None

changeGame(gameID):

Input: integer representing the game the scoreboard is displayed for.

Transition: Change the value of displayGame to the value of gameID.

Output: None

Exceptions: None

exitScoreboard():

Input: None

Transition: call the appropriate draw function (such as displayLauncher, or drawInterface of a minigame) depending on the exitScreen variable.

Output: None

Exceptions: None

readData():

Input: None

Transition: reads data for a dedicated file and store them in scores.

Output: None

Exceptions: FileNotFoundException - cannot find the file to be read, FileCannotRead - unable to read the file

writeData():

Input: None

Transition: write the values in scores to a dedicated file.

Output: the dedicated file to file system of the computer.

Exceptions: FileCannotWrite - unable to write to file

updateScore(gameID, score):

Input: first integer representing which game the score is from, second integer representing the score to be updated.



Transition: search for the first element in the appropriate array in scores(depending on the value of gameID) that is smaller than score, if not found, do nothing. Otherwise, save the score into that location in the array and move every value starting at that position to the next index.

Output: None

Exceptions: None

highScore(gameID):

Input: integer representing a mini-game

Transition: Return the highest score of the inputted mini-game (gameID).

Output: The highest score of the mini-game

Exceptions: None

## 3 MIS of Maze Generation Module

### 3.1 Interface Syntax

#### 3.1.1 Exported Access Programs

Name	In	Out	Exceptions
Cell	integer, integer	-	Invalid Input
Cell_genWalls	-	-	-
Maze	integer	-	Invalid Input
Maze_genMaze	-	-	-

### 3.2 Interface Semantics

#### 3.2.1 State Variables

cellWalls: array of integers - representing walls around the current cell

mazeWalls: array of Cell - representing the layout of the maze

#### 3.2.2 Environment Variables

None

#### 3.2.3 Assumptions

Variables should be set before trying to access them

Constructor Cell will be called before genWalls or Maze can be called

Constructor Maze will be called before genMaze can be called

### 3.2.4 Access Program Semantics

Cell(id, gridLength):

Input: two integers representing the cell ID and the maze dimensions

Transition: initializes the Cell object

Output: None

Exceptions: Invalid Input they are not positive integers

Cell\_genWalls():

Input: None

Transition: adds integers corresponding to neighbouring cells to cellWalls

Output: None

Exceptions: None

Maze(size):

Input: integer representing the size of the maze

Transition: initializes the Maze Object

Output: None

Exceptions: Invalid Input size is not a positive integer

Maze\_genMaze():

Input: None

Transition: utilizes Prim's Algorithm to randomly remove walls from the maze and manipulates mazeWalls to represent the remaining walls of the maze

Output: None

Exceptions: None

## 4 MIS of Score Tracking (Maze) Module

### 4.1 Interface Syntax

#### 4.1.1 Exported Access Programs

Name	In	Out	Exceptions
saveScore	float	-	Invalid Input
checkRank	float	integer	Invalid Input

### 4.2 Interface Semantics

#### 4.2.1 State Variables

score: float - represents the user's score once the maze is completed

#### 4.2.2 Environment Variables

None

#### 4.2.3 Assumptions

Variables should be set before trying to access them

#### 4.2.4 Access Program Semantics

saveScore(time):

Input: a float value representing the total elapsed time during the game

Transition: saves the score to the maze scores file

Output: None

Exceptions: Invalid Input if the input is not a positive float

checkRank(time):

Input: a float value representing the total elapsed time during the game

Transition: None

Output: the user's current rank based upon previous scores

Exceptions: Invalid Input if the input is not a positive float

## 5 MIS of Draw Game (Maze) Module

### 5.1 Interface Syntax

#### 5.1.1 Exported Access Programs

Name	In	Out	Exceptions
drawMaze	Maze	GUI	Invalid Input
drawCharacter	integer, integer	GUI	Invalid Input
showTime	-	GUI	-

### 5.2 Interface Semantics

#### 5.2.1 State Variables

charPos: x,y - coordinates of the character's current position

timeElapsed: float - represents the current time elapsed

#### 5.2.2 Environment Variables

keyDown: captures which key is currently being pressed down

### 5.2.3 Assumptions

Variables should be set before trying to access them

Maze must be properly initialized before drawTime can be called

### 5.2.4 Access Program Semantics

drawMaze(Maze):

Input: Maze object used to draw the layout

Transition: None

Output: draws the maze to the output window

Exceptions: Invalid Input if the object is not of type Maze

drawCharacter(startx,starty):

Input: two integers representing the coordinates to draw the character

Transition: adjusts charPos based on keyDown using Player Movement Module

Output: character is drawn according to it's current position of the maze

Exceptions: Invalid input if the integers are not of the correct coordinates

showTime(time):

Input: a float representing the current time elapsed

Transition: None

Output: a clock on the output window representing the current time elapsed

Exceptions: Invalid Input if the input is not a float or negative

## 6 MIS of Player Movement (Maze) Module

### 6.1 Interface Syntax

#### 6.1.1 Exported Access Programs

Name	In	Out	Exceptions
moveUp	-	integer, integer	-
moveDown	-	integer, integer	-
moveLeft	-	integer, integer	-
moveRight	-	integer, integer	-

### 6.2 Interface Semantics

#### 6.2.1 State Variables

charPos: int, int - representing the character's current position as coordinates (x,y)

### 6.2.2 Environment Variables

None

### 6.2.3 Assumptions

Variables should be set before trying to access them

### 6.2.4 Access Program Semantics

moveUp():

Input: None

Transition: Adjust charPos upwards (decrease y coordinate)

Output: two integers representing the new position of the character

Exceptions: None

moveDown():

Input: None

Transition: Adjust charPos downwards (increase y coordinate)

Output: two integers representing the new position of the character

Exceptions: None

moveLeft():

Input: None

Transition: Adjust charPos to the left (decrease x coordinate)

Output: two integers representing the new position of the character

Exceptions: None

moveRight():

Input: None

Transition: Adjust charPos to the right (increase x coordinate)

Output: two integers representing the new position of the character

Exceptions: None

## 7 MIS of Menu and Settings (Maze) Module

### 7.1 Interface Syntax

#### 7.1.1 Exported Access Programs

Name	In	Out	Exceptions
drawInterface	integer	GUI	-
checkEvent	float, float, boolean	integer	Invalid Input

### 7.2 Interface Semantics

#### 7.2.1 State Variables

currState: int - represents the game's current state

#### 7.2.2 Environment Variables

mousePos: the mouse/pointer's current position mouseEvent: captures a mouse event

#### 7.2.3 Assumptions

Variables should be set before trying to access them

If no event is chosen, checkEvent returns a default value 0

If currState is 0, drawInterface does not change

#### 7.2.4 Access Program Semantics

drawInterface(currState):

Input: an integer representing the game's current state

Transition: None

Output: draws the interface corresponding to the current state to the output window

Exceptions: Invalid Input if the input doesn't correspond to a game state

checkEvent(xpos, ypos, clicked):

Input: float values representing the mouse's current position on the screen and if the mouse has been clicked

Transition: determines if the current position represents a specified event

Output: integer representing the current state of the game based on the mouse

Exceptions: Invalid Input if the coordinates are not part of the window

## 8 MIS of Ball Tracking Module

### 8.1 Interface Syntax

#### 8.1.1 Exported Access Programs

Name	In	Out	Exceptions
Ball	integer, integer, integer, [integer, integer]	None	Invalid Input
ballColor	integer, integer, integer	(integer, integer, integer)	Invalid Input
checkBounds	None	None	None
scoreGoal	None	None	None
resetBall	None	None	None
drawBall	None	None	None

### 8.2 Interface Semantics

#### 8.2.1 State Variables

ballCenter\_x: int - horizontal coordinate of the ball

ballCenter\_y: int - vertical coordinate of the ball

#### 8.2.2 Assumptions

Variables should be set before trying to access them

Ball is never created outside the visible area

Top, left, right, and bottom of ball can be calculated with ballCenter\_x and ballCenter\_y

Ball is initiated before other methods are called

#### 8.2.3 Access Program Semantics

Ball(x, y, size, color, movement):

Input: integers for (x,y) position of the ball, its size, and the (x,y) speed of the ball

Transition: initializes the Ball object

Exceptions: Invalid Input if they are neither integers nor positive

ballColor():

Input: three integers each equivalent to a red, green, or blue color value

Transition: converts the three integers into a color tuple to create the ball's color

Output: color tuple based on desired ball color

Exceptions: Invalid Input if numbers are not positive integers or not in between 0 and 255

checkBounds():

Input: None

Transition: flip vertical speed at vertical bound collision, hold still at horizontal bounds

Exceptions: None

scoreGoal():

Input: None

Transition: ball at right bound is +1, ball at left bound is -1

Exceptions: Invalid Input size is not a positive integer

resetBall():

Input: None

Transition: moves the ball back to the center of the screen (after a side has scored)

Exceptions: None

drawBall():

Input: None

Transition: displays the ball to the screen

Exceptions: None

## 9 MIS of Score Tracking (Pong) Module

### 9.1 Interface Syntax

#### 9.1.1 Exported Access Programs

Name	In	Out	Exceptions
saveScore	integer, integer	[integer, integer]	Invalid Input
checkRank	[integer, integer]	integer	Invalid Input

### 9.2 Interface Semantics

#### 9.2.1 State Variables

playerScore: int - scores earned by main player

opponentScore: int - scores earned by secondary/computer player

#### 9.2.2 Assumptions

Variables should be set before trying to access them

saveScore is only called after a game is over

#### 9.2.3 Access Program Semantics

saveScore(playerScore, opponentScore):

Input: two integers representing the scores obtained by the players

Transition: converts two scores into an array



Output: a 1x2 integer array containing the two scores  
Exceptions: Invalid Input if the inputs are not positive integers

checkRank([playerScore, opponentScore]):

Input: 1x2 integer array containing the main and secondary player's score

Transition: None

Output: the games current rank based upon previous scores

Exceptions: Invalid Input if the input is not a positive integer array of the right size

## 10 MIS of Draw Game (Pong) Module

### 10.1 Interface Syntax

#### 10.1.1 Exported Access Programs

Name	In	Out	Exceptions
drawPong	Ball	GUI	Invalid Input
drawSprites	Paddle	GUI	None
showScore	scoreGoal	GUI	None

### 10.2 Interface Semantics

#### 10.2.1 State Variables

gameOver: bool - determines if game has ended or not; keeps current match running

playerOne: Paddle - movement of main player

playerTwo: Paddle - movement of secondary player

score: int - keeps track of total points earned

fps: int - keeps track of the number of times (per second) to refresh the game's state

#### 10.2.2 Access Program Semantics

drawPong(Ball):

Input: Ball object to move around

Transition: None

Output: draws the ball to the output window

Exceptions: Invalid Input if the object is not of type Ball

drawSprites(Paddle):

Input: Paddle from Player Movement (Pong) Module

Transition: None

Output: primary player's paddle from Player Movement (Pong) Module drawn on-screen

Exceptions: Invalid input if the Paddle does not yet exist

showScore(scoreGoal):

Input: scoreGoal is a notifier of score change

Transition: positive notifier means primary player scored, negative means secondary player scored

Output: two numbers shown on screen representing each player's score

Exceptions: Invalid Input if the input is not an integer

## 11 MIS of Player Movement (Pong) Module

### 11.1 Interface Syntax

#### 11.1.1 Exported Access Programs

Name	In	Out	Exceptions
Paddle	integer, integer, color	None	Invalid Input
checkBounds	None	None	None
movePaddle	None	None	None
drawPaddle	None	None	None

### 11.2 Interface Semantics

#### 11.2.1 State Variables

paddleCenter\_x: int - horizontal coordinate of a paddle

#### 11.2.2 Environment Variables

None

#### 11.2.3 Assumptions

Variables should be set before trying to access them

The four corners of a paddle can be calculated from paddleCenter\_x and paddleCenter\_y

#### 11.2.4 Access Program Semantics

Paddle(integer, integer, integer, color):

Input: 2 integers for the paddle's position and a color tuple defining the paddle's color

Transition: initializes a Paddle object

Output: None

Exceptions: None

movePaddle():

Input: None

Transition: moves the paddle (up or down) based on the keyboard input

Output: paddle's vertical coordinate increases or decreases

Exceptions: None

drawPaddle():

Input: None

Transition: recalculate the position of a paddle

Output: displays the paddle to the screen

Exceptions: None

## 12 MIS of Menu and Settings (Pong) Module

### 12.1 Interface Syntax

#### 12.1.1 Exported Access Programs

Name	In	Out	Exceptions
drawInterface	integer	GUI	-
checkEvent	float, float, boolean	integer	Invalid Input

### 12.2 Interface Semantics

#### 12.2.1 State Variables

currState: int - represents the game's current state

#### 12.2.2 Environment Variables

mousePos: the mouse/pointer's current position

mouseEvent: captures a mouse event

#### 12.2.3 Assumptions

Variables should be set before trying to access them

If no event is chosen, checkEvent returns a default value 0

If currState is 0, drawInterface does not change

### 12.2.4 Access Program Semantics

`drawInterface(currState):`

Input: an integer representing the game's current state

Transition: None

Output: draws the interface corresponding to the current state to the output window

Exceptions: Invalid Input if the input doesn't correspond to a game state

`checkEvent(xpos, ypos, clicked):`

Input: float values of the mouse's position on the screen and if the mouse has been clicked

Transition: determines if the current position represents a specified event

Output: integer representing the current state of the game based on the mouse

Exceptions: Invalid Input if the coordinates are not part of the window

## 13 MIS of Score Tracking (Flappy) Module

### 13.1 Interface Syntax

#### 13.1.1 Exported Access Programs

Name	In	Out	Exceptions
<code>saveScore</code>	integer, integer	None	Invalid Input
<code>checkRank</code>	integer	integer	Invalid Input

### 13.2 Interface Semantics

#### 13.2.1 State Variables

`playerScore`: float - total time elapsed during the game

#### 13.2.2 Assumptions

Variables should be set before trying to access them

`saveScore` is only called after a game is over

#### 13.2.3 Access Program Semantics

`saveScore(playerScore):`

Input: one float representing the score obtained by the player.

Transition: saves the score to the flappy scores file

Output: None

Exceptions: Invalid Input if the input are not positive float

`checkRank([playerScore]):`

Input: a integer value representing the user's score during the game.

Transition: None

Output: the user's current rank based upon previous scores

Exceptions: Invalid Input if the input is not a positive integer

## 14 MIS of Map Generation (Flappy) Module

### 14.1 Interface Syntax

#### 14.1.1 Exported Access Programs

Name	In	Out	Exceptions
topPipe	[integer, integer, integer]	-	Invalid Input
bottomPipe	[integer, integer, integer]	-	Invalid Input
pipeDist	-	[integer, integer, integer]	Invalid Input

### 14.2 Interface Semantics

#### 14.2.1 State Variables

topPipePos: int, int - representing the top pipe's current position as coordinates (x,y)

bottomPipePos: int, int - representing the bottom pipes's current position as coordinates (x,y)

#### 14.2.2 Environment Variables

#### 14.2.3 Assumptions

Variables should be set before trying to access them

#### 14.2.4 Access Program Semantics

topPipe([integer, integer, size]):

Input: three integer array representing the top's X and Y position, as well as the length of the pipe. Top pipe's location are centered at the top base of the pipe.

Transition: initializes the topPipe object

Output: None

Exceptions: Invalid input if size exceeds the game window

bottomPipe([integer, integer, size]):

Input: three integer array representing the top's X and Y position, as well as the length of the pipe.

Transition: initializes the bottomPipe object

Output: None

Exceptions: Invalid input if size exceeds the game window

pipeDist():

Input: None

Transition: None

Output: three integer array with randomly chosen integers for x, y and the size.

Exceptions: None

## 15 MIS of Draw Game (Flappy) Module

### 15.1 Interface Syntax

#### 15.1.1 Exported Access Programs

Name	In	Out	Exceptions
drawTopPipe	topPipe	GUI	Invalid Input
drawBottomPipe	bottomPipe	GUI	Invalid Input
drawBird	bird	GUI	Invalid Input
drawScore	score	GUI	Invalid Input

### 15.2 Interface Semantics

#### 15.2.1 State Variables

#### 15.2.2 Environment Variables

#### 15.2.3 Assumptions

Variables should be set before trying to access them

#### 15.2.4 Access Program Semantics

drawTopPipe(topPipe):

Input: topPipe object to move around

Transition: None

Output: draws the topPipe to the output window

Exceptions: Invalid input if input is not of type topPipe

drawbottomPipe(bottomPipe):

Input: bottomPipe object to move around

Transition: None

Output: draws the bottomPipe to the output window  
Exceptions: Invalid input if input is not of type bottomPipe

drawBird(bird):  
Input: bird object to move around  
Transition: None  
Output: draws the bird object to the output window  
Exceptions: Invalid input if input is not of type bird

showScore(score): Input: score is a notifier of score change  
Transition: Increases score by one  
Output: draws the current score to the output window  
Exceptions: Invalid input if input is not of type integer

## 16 MIS of Player Movement (Flappy) Module

### 16.1 Interface Syntax

#### 16.1.1 Exported Access Programs

Name	In	Out	Exceptions
Bird	[integer, integer]	None	Invalid Input
moveBird	bottomPipe	None	Invalid Input
updateBird	[integer, integer]	None	Invalid Input

### 16.2 Interface Semantics

#### 16.2.1 State Variables

birdCenter\_pos: [int, int] - x,y of the player's bird

#### 16.2.2 Environment Variables

#### 16.2.3 Assumptions

Variables should be set before trying to access them

#### 16.2.4 Access Program Semantics

Bird([integer, integer]):  
Input: two integer array representing the bird's X and Y position  
Transition: initializes the Bird object

Output: None  
Exceptions: None

moveBird():  
Input: None  
Transition: moves the bird up based on the keyboard input  
Output: bird's vertical coordinate increases  
Exceptions: None

updateBird([integer, integer]):  
Input: [integer, integer] with new bird positions  
Transition: initializes the Bird object  
Output: None  
Exceptions: None

## 17 MIS of Menu and Settings (Flappy) Module

### 17.1 Interface Syntax

#### 17.1.1 Exported Access Programs

Name	In	Out	Exceptions
drawInterface	integer	GUI	-
checkEvent	float, float, boolean	integer	Invalid Input

### 17.2 Interface Semantics

#### 17.2.1 State Variables

currState: int - represents the game's current state

#### 17.2.2 Environment Variables

mousePos: the mouse/pointer's current position  
mouseEvent: captures a mouse event

#### 17.2.3 Assumptions

Variables should be set before trying to access them  
If no event is chosen, checkEvent returns a default value 0  
If currState is 0, drawInterface does not change



#### 17.2.4 Access Program Semantics

`drawInterface(currState):`

Input: an integer representing the game's current state

Transition: None

Output: draws the interface corresponding to the current state to the output window

Exceptions: Invalid Input if the input doesn't correspond to a game state

`checkEvent(xpos, ypos, clicked):`

Input: float values of the mouse's position on the screen and if the mouse has been clicked

Transition: determines if the current position represents a specified event

Output: integer representing the current state of the game based on the mouse

Exceptions: Invalid Input if the coordinates are not part of the window