**Brick Breaker Game Plan – Cameron Miller**

**List of Classes needed**

-Player

-Ball

-Brick

**List of Functions needed**

-Draw Function

-Player Move

-Ball move

-Ball Collisions

-Make Bricks

-Brick Collisions

-Difficulty

-Write text (draw score)

-Key Press and Key Release

**Variables needed for each thing**

**Player/bouncer Variables**

playerXpos = 0

playerYpos = 0

playerXspeed = 5

**Constants**

PLAYER\_HEIGHT = 20

PLAYER\_WIDTH = 100

**Ball Variables**

ballXpos = 0

ballYpos = 0

ballXspeed = 5

ballYspeed = 5

**Constants**

BALL\_WIDTH = 10

BALL\_HEIGHT = 10

**Brick variables**

**-**BrickRow = 1

-BrickTotal = 10 \* brickRow

-BrickCount = 0

-Bricks = [] (array)

-BrickxPos = 0

-BrickyPos = 0

**Constants**

BRICK\_WIDTH = 50;

BRICK\_HEIGHT = 25;

**Button Press variables**

leftKeyPressed = false

rightKeyPressed = false

**Constants**

LEFT\_KEY = 65 (number is key code)

RIGHT\_KEY = 68

**Other variables**

Score = 0

Difficulty = ‘’

**Functions in detail**

**Function to draw everything in game**

Function draw (x, y, w, h, c) {

canvasContext.fillStyle = c

canvasContext.fillRect(x, y, w, h)

}

**Function to write text in game like score which gets all style, colour and position of where to put it**

Function Write text (style, colour, text, x, y)

canvasContext.font = style

canvasContext.fillStyle = colour

canvasContext.fillText (text, x, y)

}

**Function to see if the key has been pressed**

Function key Press {

If (key Code == LEFT\_KEY) {

leftKeyPressed = true

}

If (key Code == RIGHT\_KEY) {

rightKeyPressed = true

}

}

**Function to see if the key has been released**

Function key Release {

If (key Code == LEFT\_KEY) {

leftKeyPressed = false

}

If (key Code == RIGHT\_KEY) {

rightKeyPressed = false

}

}

**Function that asks player the game difficulty using prompt**

Function difficulty {

**If game difficulty is not set or is entered wrong using following it will re prompt**

while (Difficulty !== 'easy' && Difficulty !== 'hard' || Difficulty = '' || Difficulty = null) {

Difficulty = prompt ("Please enter your difficulty Easy or Hard”) || '';

Difficulty = Difficulty.toLowerCase () } **Sets what is entered to lowercase**

**If Difficulty is easy or hard it changes the amount of brick rowswhich changes total bricks as it is 10\*brickRow**

if (Difficulty = 'easy') {

brickRow = 3

}

if (Difficulty = "hard") {

brickRow = 5

}

}

**Classes**

**Player class which is what the paddle is where the player controls it**

Class Player {

Constructor(x, y, w, h, c, xSpeed) {

this.x = x

this.y = y

this.w = w

this.h = h

this.c = c

this.xSpeed = xSpeed

}

**The following uses the function earlier to draw the player using what was set up.**

Draw Player () {

canvasContext.fillStyle = this.c

canvasContext.fillRect (this.x, this.y, this.w, this.h)

}

**The following Uses what the keypress function earlier did to move player**

Player Move () {

if (rightKeyPressed = true) {

this.x = this.x + this.xSpeed

**If the player is at the end it will move it back so it doesn’t go past the screen.**

if (this.x > canvas.width - this.w){

this.x = canvas.width - this.w

}

}

**Same for the Left movement**

if (leftKeyPressed = true) {

this.x = this.x - this.xSpeed;

if (this.x < 0){

this.x = 0;

} } }

**Brick Class**

Class Brick {

Constructor(x, y, w, h, c) {

this.x = x

this.y = y

this.w = w

this.h = h

this.c = c

}

**Same as draw Player uses earlier function to draw the brick**

Draw Brick () {

canvasContext.fillStyle = this.c

canvasContext.fillRect (this.x, this.y, this.w, this.h)

}

}

**Function that makes the bricks from the class**

Function make Bricks ()

var gap = 10; **Private variable for the gap between bricks**

var brickCol = brickCount % 10 **The column will set back to 1 everytime it reaches 10 to start a new row using mod % which also what happens below**

If (brickCount % 10 = 0 & brickCount > 0) {

brickRows++ }

BrickxPos = brickCol \* (BRICK\_WIDTH + gap) + gap;

BrickyPos = brickRows \* (BRICK\_HEIGHT + gap) + gap **Sets the x and y position by \* by current col and width of each to know where to place**

var b = new Brick(xPos, yPos, BRICK\_WIDTH, BRICK\_HEIGHT, colour) **(uses actual variables to draw the brick)**

Bricks.push(b) **pushed the brick into an array from the variables earlier**

brickCount++ **adds to the brick count so knows when to start new row**

}

**Ball Class**

Class Ball {

Constructor(x, y, w, h, c, xSpeed, ySpeed) {

this.x = x

this.y = y

this.w = w

this.h = h

this.c = c

this.xSpeed = xSpeed

this.ySpeed = ySpeed

}

**Draws the ball like the bricks and player**

Draw Ball() {

canvasContext.fillStyle = this.c

canvasContext.fillRect(this.x, this.y, this.w, this.h)

}

**Ball Move function**

Function ball Move () {**this is what moves the ball when it bounces off walls and keeps it moving**

this.x = this.x + this.xSpeed

this.y = this.y + this.ySpeed

If (this.x < 0) {**changes direction if touching left**

this.xSpeed = this.xSpeed \* -1

}

If (this.x + this.w > canvas.width) {**changes the ball direction if touching the right**

this.xSpeed = this.xSpeed \* -1

}

If (this.y < 0) {**changes the ball direction if touching top wall**

this.ySpeed = this.ySpeed \* -1

}

If (this.y > canvas.height - BALL\_HEIGHT) {stops **ball if touching bottom cause player to lose**

this.ySpeed = 0

this.xSpeed = 0

}

}

**Function that allows ball to bounce when touching the player**

Function Player Collide {

If (this.x > bouncer.x && this.x < bouncer.x + bouncer.w &&

this.y + this.h > bouncer.y && this.y < bouncer.y + bouncer.h) {

this.ySpeed = this.ySpeed \* -1 **Changes direction of ball when touching player**

this.xSpeed = this.Xspeed \* -1

}

**Function that allows ball to bounce when touching the bricks in the array**

Function Brick Collide {

If (this.y < brick.y && this.x + this.w > brick.x) {brick **top bounce**

this.ySpeed \*= -1 **Changes direction**

this.xSpeed \*= 1

Delete Bricks [] **Deletes brick from array after hit**

Score++ **Adds to score for player**

}

If (this.y + this.h > brick.y + brick.h && this.x < brick.x + brick.w) {**bottom brick bounce**

this.ySpeed \*= -1

this.xSpeed \*= 1

Delete Bricks []

Score++

}

If (this.x < brick.x && this.y < brick.y + brick.h) {left **bounce off brick horizontal move**

this.xSpeed \*= -1

Delete Bricks []

Score++

}

If (this.x + this.w > brick.x + brick.w && this.y < brick.y + brick.h) {right **edge bounce off brick horizontal move**

this.xSpeed \*= -1

Delete Bricks []

Score++

}

}