**Breakout – Phase 3 – Interactive Phase**

**Part I. Adding Timers**

Let’s start by adding two timers from the design phase. Call the timers ballTimer and paddleTimer. The ballTimer should have an interval of 25 and should not be enabled (set to false). The paddleTimer should be set with an interval of 10 and should be enabled (set to true).

**Part II. Form1\_KeyDown Event**

The Form1\_KeyDown event will need to be created. We need four “IF” statements in the KeyDown event.

* The Left Arrow key does nothing more than change the paddleDir variable to “left”.
* The Right Arrow key changes paddleDir to “right”.
* The space bar (Keys.Space) will call the resetGame() sub and start the ballTimer. However, it will only do this if the ballTimer is not enabled. (so game can’t reset while you are playing).
* The R key will also call resetGame() if the ballTimer isn’t enabled. (To replay once the game is over)

**Part III. Form1\_KeyUp Event**

* Only thing needed here is paddleDir is changed to “nothing”.

**Part IV. BallTimer\_Tick Event**

* This controls the movement of the ball. We have a variable to control the horizontal direction of the ball (ballHorizDir) and another to control its vertical direction (ballVertDir). We start this sub with four if statements. The ball always moves diagonally, so we need to handle each of the four diagonal directions the ball could potentially move:
* For each of these four possible directions, move the ball 5 pixels vertically and horizontally with each tick of the ballTimer timer.
* After the four IF statements, call the checkCollisions sub. checkCollisions() This will be used to check to see, after moving the ball, if it hits a block, or any of the walls.
* Finally, add one last IF statement that checks to see if the Y Location of the ball is > 660. This will mean the ball missed the paddle and the game is over. If this happens, the following things should occur:
  + Stop the ballTimer.
  + Messagebox that the game is over.

**Part V. paddleTimer\_Tick Event**

Three IF statements here:

* The first IF statement will move the paddle to left 5 pixels if paddleDir = “left”. We should also make sure the X location of the paddle can’t move into the left wall. This is similar to programs where we prevented an object from moving off the screen. (ex: Mario)
* The second IF statement will move the paddle to right 5 pixels if paddleDir = “right”. We should also make sure the X location of the paddle can’t move into the right wall.
* The third IF statement will move the ball to the center top of the paddle, if the ballTimer isn’t enabled. This will also us to move the ball with the paddle before we press the space bar to start the game. \*Hint the location of the ball will be based on the paddle’s X and Y location.

**Part VI. Public Sub CheckCollisions()**

This is where we will check for collisions between the ball and all 98 blocks, as well as the three walls and the paddle.

Use the following code to check for collisions between the ball and the blocks. You will notice when the ball hits a block, the block is sent to location (1000, 1000), which is off the board. It also has a counter that gets increased each time a collision occurs.

For Me.x = 1 To 98

If ball.Bounds.IntersectsWith(block(x).Bounds) Then

block(x).Location = New Point(1000, 1000)

blockCounter = blockCounter + 1

checkClear()

FlipVertical()

Exit For

End If

Next

We also need four more IF statements to check to see if the ball has hit the 3 walls or the paddle. Each of these “IF” statements should change the appropriate ballDir variable. Below is the code for the leftWall. You write the IF statements for the rightWall, topWall and paddle collisions.

If ball.Bounds.IntersectsWith(leftwall.Bounds) Then

ballHorizDir = "right"

End If

**Public Sub FlipVertical()**

* This is a simple IF…THEN…ELSE statement that will change the vertical direction of the ball.
* If the ball was moving up, it should change to moving down and vice versa.
* As a reminder the only thing we have to do to change the vertical direction of the ball is to change the ballVertDir variable.
* This is similar to our Lightbulb program.

**Public sub CheckClear()**

* In this sub, we simply check to see if the blockCounter variable is 98. If it is, we should call the placeBlocks sub, which will reposition all 98 blocks back on the board.
* We should also reset the blockCounter to 0
* If you want, you can reduce the interval of the ballTimer to speed up the movement of the ball. (Not required)

**Public Sub ResetGame()**

* Set the blockCounter to 0
* Call the placeBlocks() sub.

**Extra Credit Opportunities**

* **Scoring** - You can implement scoring for extra credit. Add labels (built from code) above to track the current score and high score.
* **Add different angles to the ball’s movement** – In our game, the ball will always move at a 45 degree angle. In the real game, if the ball hits near the end of the paddle, it will rebound at a sharper angle (X movement is more than the Y movement each tick). Try to implement something like this.
* **Powerups** – Try to add power ups that randomly spawn. Possible powerups could be making the paddle bigger. Allowing the paddle to fire a bullet towards any block above it, or whatever you can come up with. The powerups could drop from the top of the screen and are collected when the paddle hits it.