Part 3: Putting it all together

Objective: Add inner walls to build maze

Copy existing PictureBoxes and paste to create new walls. These walls will be the mazes’ inner walls.

Position anywhere within the 4 outer walls to create maze structure.

Note the name of each new PictureBox, e.g. PictureBox15, PictureBox16, PictureBox17 etc.

Each new inner wall will need collision detection, unless it’s a fake wall (walls that are intended to allow the player through but look like a normal wall).

Copy existing collision detection code for walls and paste new PictureBox number in.

If PictureBox1.Bounds.IntersectsWith(PictureBox2.Bounds) Then

PictureBox1.Top = 53

PictureBox1.Left = 125

End If

Becomes:

If PictureBox1.Bounds.IntersectsWith(PictureBox15.Bounds) Then

PictureBox1.Top = 53

PictureBox1.Left = 125

End If

Repeat for all walls having collision detection.

Fake and Invisible Walls

This game will have two fake walls and one invisible wall. The fake walls will look like a normal wall, but won’t have a collision detection. This will allow the player to pass through. These are to conceal secrets or alternate routes. The invisible wall is a hidden danger for the player. This wall can’t be seen and does have collision detection, so the player will only know it’s there when they move into it. At this time, they will be transported to the start position so will have to remember it’s there to avoid going into it again.

Save all and rebuild solution.

Start.

TEST: Move PictureBox1 into new wall PictureBoxes. When one of the edges of PictureBox1 touches an edge of a new wall PictureBox, the If statement becomes true and PictureBox1 will re-position to the co-ordinates.

RESULT: Pass.

TEST: Move PictureBox1 into fake wall PictureBoxes. As there is no collision detection with these walls and no code added at all, the wall will inherit its default settings. The player should be able to pass through these walls and there be no affect.

RESULT: Pass.

TEST: Move PictureBox1 into new invisible wall PictureBox. When one of the edges of PictureBox1 touches an edge of the new invisible wall PictureBox, the If statement becomes true and PictureBox1 will re-position to the new co-ordinates.

RESULT: Pass.

TEST: Move PictureBox1 into existing PictureBoxes to make sure new code doesn’t affect old code. When one of the edges of PictureBox1 touches an edge of an existing PictureBox, the If statement becomes true and PictureBox1 will re-position to the new co-ordinates.

RESULT: Pass.

Objective Met

Future Consideration: Moving Walls.

Objective: Add Cursed Collectables

Cursed collectables are items the player will be fooled into collecting because of their treasure like appearance, and because collecting treasure is a long-established feature of gaming.

What actually happens is the player will be transported back to the start of the game and suffer a point deduction (when the scoring system is implemented).

These items are actually traps and do not benefit the player in anyway. They will employ the same collision detection the walls use.

'JEWELS

If PictureBox1.Bounds.IntersectsWith(PictureBox6.Bounds) Then

PictureBox1.Top = 53

PictureBox1.Left = 125

End If

If PictureBox1.Bounds.IntersectsWith(PictureBox8.Bounds) Then

PictureBox1.Top = 53

PictureBox1.Left = 125

End If

If PictureBox1.Bounds.IntersectsWith(PictureBox9.Bounds) Then

PictureBox1.Top = 53

PictureBox1.Left = 125

End If

I’ll add the code for point deduction to these if statements later.

I have used comments in the code to make it a bit easier to read and navigate through. It will also make it easier for others users looking at the code to understand.

Save all and rebuild solution.

Start.

TEST: Move PictureBox1 into Jewel PictureBoxes. When one of the edges of PictureBox1 touches an edge of a Jewel PictureBox, the If statement becomes true and PictureBox1 will re-position to the co-ordinates. N.B. PictureBox6 is only accessible if the player finds and goes through the fake wall (PictureBox18).

RESULT: Pass.

Objective Met

Future Consideration: Adding collectables that change gameplay e.g. powerups, or increase score e.g. coins.

Objective: Add means of winning the game

In Design view add timer from Toolbox.

In code view copy if statement for wall collision detection and remove instructions inside statement.

Change .IntersectsWith… to .IntersectsWith(PictureBox7.Bounds) touching PictureBox7 will complete and end the game. I have used a cartoon idol of the Incan God Viracocha to represent the goal.

Add the following code within the if statement:

MsgBox("Your Score Is:" & "" & score)

Timer1.Stop()

Create a global variable called score so we can use it to for our score feature later.

Place it within Public Class Form1, but outside of Private Sub Form1\_KeyDown(sender As Object, e As KeyEventArgs) Handles MyBase.KeyDown

So:

Public Class Form1

Dim score As Integer

Private Sub Form1\_KeyDown(sender As Object, e As KeyEventArgs) Handles MyBase.KeyDown

Save all and rebuild solution.

Start.

TEST: Move PictureBox1 into PictureBox7. When one of the edges of PictureBox1 touches an edge of PictureBox7, the If statement becomes true and a message box will appear, display our text message and stop the timer.

RESULT: Fail. Displays message box with score at the time of touching the goal. The timer didn’t stop until I clicked ok on the message box.

Try placing the Timer1.Stop() code above the MsgBox("Your Score Is:" & "" & score) code. When this runs it will carry out Timer1.Stop() then MsgBox("Your Score Is:" & "" & score). You won’t notice a delay between the timer stopping and the message box appearing as there’s no need for the program to wait after Timer1.Stop().

Save all and rebuild solution.

Start.

TEST: Move PictureBox1 into PictureBox7. When one of the edges of PictureBox1 touches an edge of PictureBox7, the If statement becomes true and stops the timer. The message box then appears and displays our text message and score.

RESULT: Pass.

N.B. The game will still run because I assumed Timer1.Stop() would end the game as well as stop the timer. This isn’t the case. I will have to find a command to stop the game, add it to the code block, and then test to see if it works as expected.

Objective Met

However, game should close after clicking ok on message box. See Objective: Add end game command after winning game or losing game to solve this issue.

Objective: Add timer for scoring, point deduction and lose game when score reaches 0

Add label for scoring and rename to lblScore.

Change Text to display from “Label1” to “1000”.

Place in top right corner of form.

Select Timer1 and change Interval from 100 to 1000.

Set Dim score As Integer = 1000

Select Timer1 and click on Lightning bolt to open Events. Next to Tick mouse click and press enter.

Opens new code block.

Private Sub Timer1\_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick

End Sub

Add score = score – 10 inside this new block.

Timer needs to start when program runs.

Click on Form1, Events (lightning bolt), Load – click and enter.

Generates following block:

Private Sub Form1\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

End Sub

Add Timer1.Start() within this block so when we run the program, this will initialise the timer.

New Code looks like:

Private Sub Timer1\_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick

score = score - 10

End Sub

Private Sub Form1\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Timer1.Start()

End Sub

Score needs to decrease while game is being played.

Add following code inside the Timer1\_Tick block:

lblScore.Text = CStr(score)

We want the game to stop if the timer reaches 0 so add underneath:

If score = 0 Then

Timer1.Stop()

MsgBox("You Lose!!!")

End If

When the player hits an object with collision detection, they should also incur a score penalty. This will accelerate the score decline towards 0.

Create a new function and place below Private Sub Form1\_Load

This function will be called points() and inside the function will be score = score – 10. This means whenever points() is called, whatever the current score is will be reduced by 10.

This new block will look like:

Function points()

score = score - 10

End Function

points() can be used for any object that wants a point penalty when the player collides with it. It can go in the existing if statements for collision detection.

The code for jewels for example will now look like:

'JEWELS

If PictureBox1.Bounds.IntersectsWith(PictureBox6.Bounds) Then

points()

PictureBox1.Top = 53

PictureBox1.Left = 125

End If

If PictureBox1.Bounds.IntersectsWith(PictureBox8.Bounds) Then

points()

PictureBox1.Top = 53

PictureBox1.Left = 125

End If

If PictureBox1.Bounds.IntersectsWith(PictureBox9.Bounds) Then

points()

PictureBox1.Top = 53

PictureBox1.Left = 125

End If

Save all and rebuild solution.

Start.

TEST: Score to decrease by 10 with every passing second. Message box to appear when score reaches 0 and display message.

RESULT: Pass.

TEST: Hold down direction key so player repeatedly hits a wall with a point penalty. “If score = 0 Then” if statement should activate when score reaches 0, stopping play.

RESULT: Fail. “If score = 0 Then” if statement does not activate when score reaches 0 and play continues. If the key is held down the score goes into a negative range and the timer continues to increase the negative score.

I’ll try an if statement to stop incrementing the score if it equals 0.

Add an if statement inside the points() function. Enclose score = score – 10 within an if statement that is true only if the score is greater than 0. So:

Function points()

If score > 0 Then

score = score - 10

End If

End Function

Save all and rebuild solution.

Start.

TEST: Score to decrease by 10 with every time an object with collision detection is hit.

RESULT: Pass.

TEST: Hold down direction key so player repeatedly hits a wall with a point penalty. “If score = 0 Then” if statement should activate when score reaches 0, stopping play.

RESULT: Pass.

N.B. The game will still run because I assumed Timer1.Stop() would end the game as well as stop the timer. This isn’t the case. I will have to find a command to stop the game, add it to the code block, and then test to see if it works as expected.

Objective Met

However, game should close after clicking ok on message box. See Objective: Add end game command after winning game or losing game to solve this issue.

Objective: Add end game command after winning game or losing game

I need a command to end the game after the player wins or loses. I am assuming the code will go inside the winning and losing if statements:

If PictureBox1.Bounds.IntersectsWith(PictureBox7.Bounds) Then

Timer1.Stop()

MsgBox("Your Score Is:" & "" & score)

???

End If

And

If score = 0 Then

Timer1.Stop()

MsgBox("You Lose!!!")

???

End If

After researching online, Close() seems to be a clean way to stop and close a program.

I’ll add this after the MsgBox code so it doesn’t close until after the user has clicked ok on the message box.

If PictureBox1.Bounds.IntersectsWith(PictureBox7.Bounds) Then

Timer1.Stop()

MsgBox("Your Score Is:" & "" & score)

Close()

End If

And

If score = 0 Then

Timer1.Stop()

MsgBox("You Lose!!!")

Close()

End If

Save all and rebuild solution.

Start.

TEST: Win game to activate win if statement. Program should close after user clicks ok on the win message.

RESULT: Pass.

TEST: Lose game to activate lose if statement. Program should close after user clicks ok on the lose message.

RESULT: Pass.

Objective Met

Future Consideration: Instead of closing the program after a win or lose, give the player a choice to restart or close. E.g. “Hit F1 to restart” & “Hit F2 to close”.