Faculty of Information and Communication Technologies

Higher Education Division

Game Development Tutorial

Ping Pong

Quick Start Guide

7 November, 2008

SWINBURNE UNIVERSITY OF TECHNOLOGY

Ping Pong – Quick Start

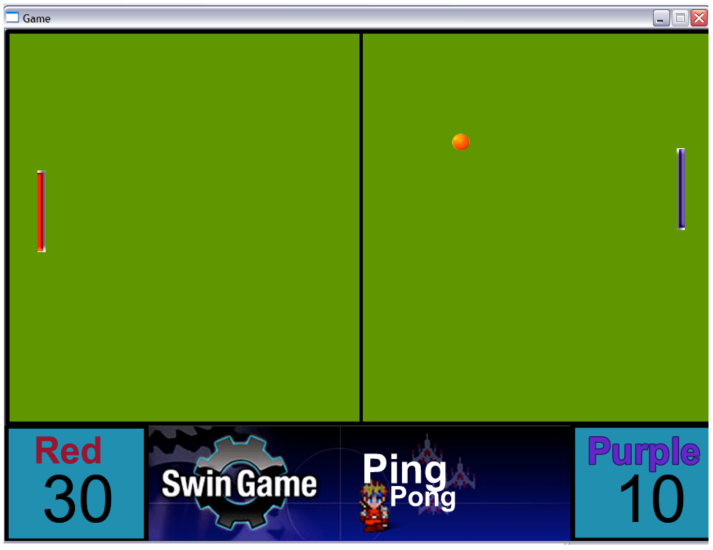
In this session you will experience software development by working on a small computer game called **Pong**. Did you know that *Pong* was the first video game to achieve widespread popularity in both arcade and home console versions? Pong launched the initial boom in the video game industry, so we think it's a great place to start writing your own games.

To build Pong, we will use Visual Basic .NET and the SwinGame SDK. Visual Basic is a computer programming language that can be used to create a variety of programs, including games. The SwinGame SDK is a games development kit that was made by the Professional Software Development group at Swinburne University. You can find out more about it at http://www.swingame.com.

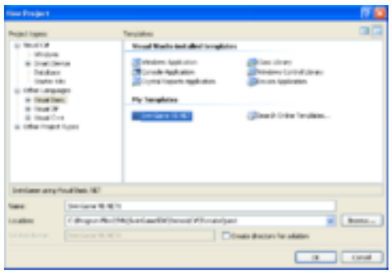
For this Quick Start you will begin with a partly working game and add some missing features. There is a more complete version of this worksheet that goes through all the steps of building this game from scratch.

Finished Game

Below is a picture of how the game will look after we have finished. It's a two player game where each player controls a paddle. The ball is served and each player moves their paddle to return the ball. If a player misses their opponent scores a point.

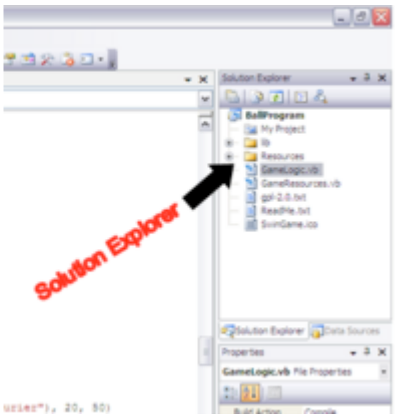


Getting Started



Before we can start creating the game we need to make sure that we have got the SwinGame project setup correctly. Follow these steps to get a "Hello World" program working with SwinGame.

1. Open Visual Studio, and click File > New > Project.
2. Select the Visual Basic/Basic Language and Click the SwinGame VB.NET Project Template. Click OK.
3. Select File - Save. Enter "**HelloWorld**" as the file name, and choose the location it is to be saved and click OK.
4. In the Solution Explorer on the right hand side of the screen double click on "GameLogic". You should be presented with the following code:



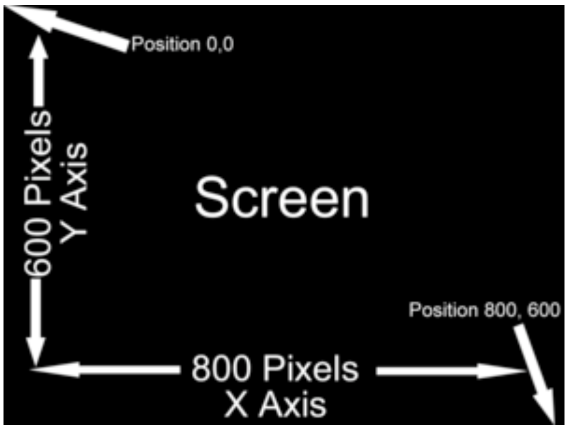
```
Module GameLogic

Public Sub Main()
  'Opens a new Graphics Window
  Core.OpenGraphicsWindow("Game", 800, 600)

  'Open Audio Device
  Audio.OpenAudio()

  'Load Resources
  LoadResources()

  'Game Loop
  Do
    'Clears the Screen to Black
    SwinGame.Graphics.ClearScreen()

    Graphics.FillRectangle(Color.Red, 20, 200, 200, 100)
    Graphics.FillRectangle(Color.Green, 220, 200, 200, 100)
    Graphics.FillRectangle(Color.Blue, 420, 200, 200, 100)

    Text.DrawText("Hello World", Color.Red, GameFont("Courier"), 20, 310)
    Text.DrawText("Hello World", Color.Green, GameFont("Courier"), 220, 310)
    Text.DrawText("Hello World", Color.Blue, GameFont("Courier"), 420, 310)

    Text.DrawFramerate(0, 0, GameFont("Courier"))
    Text.DrawText("Hello World", Color.White, GameFont("ArialLarge"), 50, 50)

    'Refreshes the Screen and Processes Input Events
    Core.RefreshScreen()
    Core.ProcessEvents()

  Loop Until SwinGame.Core.WindowCloseRequested() = True

  'Free Resources and Close Audio, to end the program.
  FreeResources()
  Audio.CloseAudio()
End Sub

End Module
```

The purpose of this code is to create a starting point for your project. If you press the "Start Debugging" button at the top of the screen (looks like a green arrow, F5 works too) you will see what it does. Have a look, then close the window.

You can basically read what it is doing
`'comments are in green with a single quote at the start like this sentence`

See if you can find the parts of the code that are doing the following things:

1. Loading all the resources you need for the game
2. Opening up a blank screen 800 pixels wide (x axis) by 600 tall (y axis).
3. Start a Loop, in which it:
   a. Draws 3 filled rectangles setting their sizes and different colours width height and position
   b. Putting the game's "Frame rate" up on the screen in the top right hand corner
   c. Writing the words "Hello World" 3 times in different colours below the rectangles
   d. Writing the words "Hello World" once at the top in large letters
4. Telling the loop to stop when someone closes the window



Exercise 1

To help your understanding of what is going on in this code do the following:

1. Change the big "Hello World" text to your name
2. Move the Red Rectangle so it is in the bottom right hand corner
3. Make the Red Rectangle twice as big

You should have now completed your "Hello World" so we can start work on the Pong game.

Starting Ping Pong

In this quick start we will be working with a partly completed version of the game. Download this from the web site and then open the PongPong solution (the PingPong.sln file).

Please Note: When you are editing the program there are some *'TODO:* items that point out the locations of the missing code. So keep an eye out for them when you need to make the changes.

The game should run already so click the "Start Debugging" button to see what we have. It doesn't really do much at the moment. You need to make some changes… so lets get started!

Part 1 Drawing the background

The background is missing... lets fix that first.

Open up the GameLogic.vb file from the Solution Explorer (it's on the right usually, double click the file to open it). Locate the **DrawGame** procedure it starts with *Sub DrawGame*. This code is called each time the screen is drawn and at the moment it isn't drawing the background. There is a TODO here so replace it with the code highlighted below.

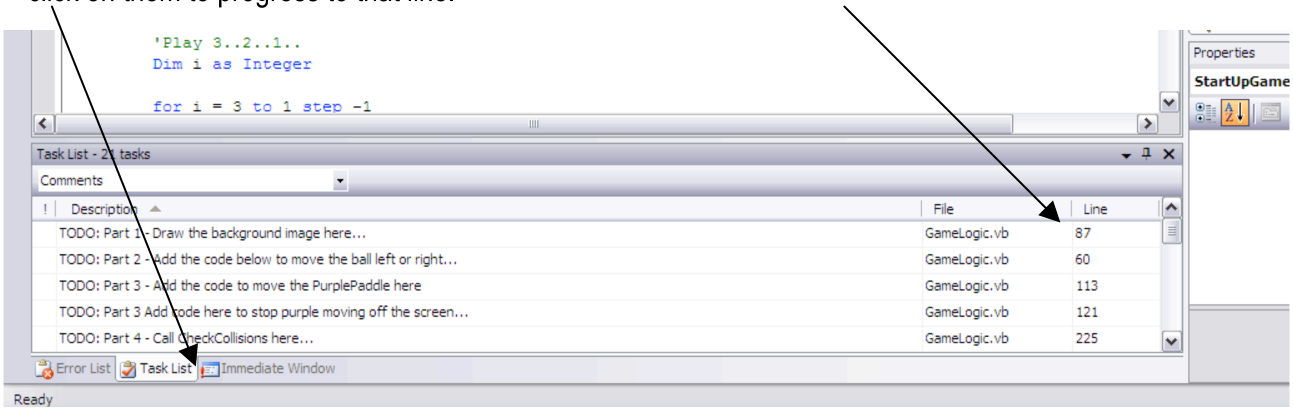
```
  Sub DrawGame()
    …
    Graphics.DrawBitmap(GameImage("table"), 0, 0)
    …
  End Sub
```

This code tells SwinGame to draw the "table" image at the top left corner (0,0) of the screen. As the image is the same size as the window (800,600) it takes up the whole background.

Hit F5 and see the image appear drawn as the background… When you run the game now it looks a lot better, but the ball isn't moving!

Hint…

To help you find the section of code you need to edit various TODO: comments have been made, if you switch to the "**Task List**" at the bottom of the screen you can see a list of all the different TODOs and you can double click on them to progress to that line.



```
        'Play 3..2..1..
        Dim i as Integer

        for i = 3 to 1 step -1
```

Task List - 21 tasks

Comments

!	Description	File	Line
	TODO: Part 1 - Draw the background image here…	GameLogic.vb	87
	TODO: Part 2 - Add the code below to move the ball left or right…	GameLogic.vb	60
	TODO: Part 3 - Add the code to move the PurplePaddle here	GameLogic.vb	113
	TODO: Part 3 Add code here to stop purple moving off the screen…	GameLogic.vb	121
	TODO: Part 4 - Call CheckCollisions here…	GameLogic.vb	225

Part 2 Moving Sprites: Getting the Ball Moving

In SwinGame things that move are called *Sprites*. There is already the code needed to create and initialise the Sprites used in the Game, but the ball has been told not to move. So we need change the starting movement of the Ball so it moves either left or right (randomly). This is set in the **StartUpGame** code. In order to get our ball moving edit the code in **StartUpGame** to set the movement of the ball so that it appears as shown below:

```
  …
Ball.Movement.Y = 0
Ball.Movement.X = 0

  ball.Movement.Y = Rnd() * 2

  'Send ball... left or right
  If Rnd() < 0.5 Then
    ball.Movement.X = -2.5
  Else
    ball.Movement.X = 2.5
  End If

  'Position of RedPaddle
  …
```

This code sets the ball moving either left or right, with an equal chance of going either way. The ball is moved by the code in **UpdateBall**, but for the moment we should have done enough to get the ball moving. Run the game and check it out.

Note: Rnd() generates a random number between 0 and 1 (like 0.126, or 0.572).

Part 3 Keyboard Input

Now we need to get the key input working. If you check the A and Z keys they already move the left paddle. The code for this is in the **HandleInput** procedure. When the user holds down the A key it will move the paddle sprite up at a rate of 2 pixels per loop. If the user holds down the Z key it will move the paddle sprite down at a rate of 2 pixels per loop. You can change these numbers if you want it to move faster or slower.

Exercise – Getting Purple Moving

Add the code to make the PurplePaddle go up and down using “Up” and “Down” arrow keys, and the code to keep the paddle on the screen.

```
Sub HandleInput()
  'Clear Movement
  redPaddle.Movement.X = 0
  redPaddle.Movement.Y = 0
  purplePaddle.Movement.X = 0
  purplePaddle.Movement.Y = 0

  'moves the paddle when user presses left and right key
  If Input.IsKeyPressed(SwinGame.Keys.VK_A) Then
    redPaddle.Movement.Y = -2
  End If

  If Input.IsKeyPressed(SwinGame.Keys.VK_Z) Then
    redPaddle.Movement.Y = 2
  End If

  If Input.IsKeyPressed(SwinGame.Keys.VK_UP) Then
    purplePaddle.Movement.Y = -2
  End If

  If Input.IsKeyPressed(SwinGame.Keys.VK_DOWN) Then
    purplePaddle.Movement.Y = 2
  End If

  'Move Paddles
  Graphics.MoveSprite(purplePaddle)
  Graphics.MoveSprite(redPaddle)

  'stops the paddles moving off the screen
  If redpaddle.Y < FIELD_TOP Then
    redpaddle.Y = FIELD_TOP
  End If

  If redpaddle.Y + redPaddle.Height > FIELD_BOTTOM Then
    redpaddle.Y = FIELD_BOTTOM - redPaddle.Height
  End If

  If purplepaddle.Y < FIELD_TOP Then
    purplepaddle.Y = FIELD_TOP
  End If

  If purplepaddle.Y + purplepaddle.Height > FIELD_BOTTOM Then
    purplepaddle.Y = FIELD_BOTTOM - purplepaddle.Height
  End If

End Sub
```

Part 4 Sprite Collisions

Although it is fun watching a ball bounce aimlessly around the screen it would be nice if we could interact with it. To do this we need to tell our program to look out for a collision, namely the collision between the ball and the paddles. If this collision occurs then obviously we need to tell the program to bounce the ball off in the other direction the same as if we had hit the bottom or top by reversing the Y axis of the path of the ball.

We need to call the **CheckCollision** procedure that checks if the ball has collided with either of the paddles and acts accordingly. Find the game loop in **Main** and add in the call to *CheckCollisions*, the code for this is shown below:

```
'Game Loop
  Do
    UpdateBall()
    HandleInput()
    CheckCollisions()

    DrawGame()

    'Refreshes the Screen and Processes Input Events
    Core.RefreshScreen()
    Core.ProcessEvents()

  Loop Until SwinGame.Core.WindowCloseRequested() = True
```

Run the game and you should be able to hit the ball back and forth.

Part 5 Playing Sounds

We want to add a sound for when the ball bounces off the outside walls and a different one when it hits the paddle.

Open up the GameLogic source as we need to tell the program when to play the sounds. The *bounce* sound should play when the ball hits one of the walls. Locate the **UpdateBall** procedure. This is the one that moves the ball and it contains some **if** statements that check if the ball has hit a wall. All we need to do now is find them and add a line of code to tell it to play the sound as well as changing the direction of the ball. Add the missing line of code to the body (between the "if" and "End If") of the four if statements, so for example one of them will look like this now:

```
    If Ball.Y < FIELD_TOP Then
        Ball.Y = FIELD_TOP
        Ball.Movement.Y = -ball.Movement.Y
        Audio.PlaySoundEffect(GameSound("bounce"))
    End If
```

Once that is added to **all four** if statements then hit "Start Debugging" and test it out. It should play the "bounce" sound every time the ball hits an outside border.

Exercise – The crowd goes wild

Play the *backwall* sound if the ball hits the left or right hand wall. To do this use the following code:

```
Audio.PlaySoundEffect(GameSound("backwall"))
```

If you want some background music in your game playing in loop it is almost the same process as we used when we added the other sounds. Find the Music TODO in the **Main** procedure and replace it with the following line:

```
    Audio.PlayMusic(GameMusic("amb"))
```

Then you will have some nice ping pong noise looping in the background. You can play any MP3 file so you could use a song or any other MP3 files for your music.

Part 6 Adding Spin and Acceleration

Now we are going to make it possible to "spin" the ball as it comes off the paddles. Locate *CheckCollisions* code and change it so that it appears as shown below. You will need to add in the code for both the red and purple paddles, only the red paddle code is shown below.

```
Sub CheckCollisions()
  'Ball collision with paddles
  If Physics.HaveSpritesCollided(Ball, RedPaddle) Then
    Audio.PlaySoundEffect(GameSound("hit"))
    PerformCollision(ball, RedPaddle)

    'Add spin (30% of paddle movement)
    ball.Movement.Y = ball.Movement.Y + 0.3 * RedPaddle.Movement.Y

    'Accelerate by 5%
    ball.Movement.Y = ball.Movement.Y * 1.05
    ball.Movement.X = ball.Movement.X * 1.05
  End If
  …
End Sub
```

Part 7 Scoring

So lets set it up so when the red player misses the ball the purple player gets 1 point and vice versa. The first thing we need to do is declare two variables (dim) as Integers (integers are just whole numbers like 1 and 50 but not 1.25). These need to be added to the top of the GameLogic file just below the other variables, so scroll to the top of the file and the missing two lines shown in the code here.

```
Private RedPaddle As Sprite
Private ScoreRed As Integer
Private ScorePurple As Integer

Sub CheckCollisions() …
```

Next we need to initialise these to 0; the score each player will start with. This needs to be done in the *Main* procedure. You will need to scroll down to the botton of the file and insert the missing code just below the *LoadResources* call.

```
LoadResources()

ScoreRed = 0
ScorePurple = 0

'Create balls and paddles
…
```

Next we just need to tell it to add to the opponents score when the ball is missed. You will remember our code already knows when the ball hits any of the walls in *UpdateBall*. We just need to find out when it hits the left or right walls and when it does add 1 to ScoreRed or ScorePurple depending on if it is the left or the right border it has hit. The code for adding 1 to the score is simple, lets say the ball hits the right border then red would get 1 point. The code for this is:

```
ScoreRed = ScoreRed + 1
```

Find the correct location for this in the *UpdateBall* procedure. Remember to add it for both the Red and Purple players.

Now to display that score on the screen below the word “Red” you need to add the following code to the *DrawGame* procedure, add it near the end of that procedure (you will remember it from the "Hello World" tutorial). The new *DrawGame* is shown below.

```
Sub DrawGame()
    'Drawing the game elements
    Graphics.DrawBitmap(GameImage("table"), 0, 0)
    Graphics.DrawSprite(ball)
    Graphics.DrawSprite(PurplePaddle)
    Graphics.DrawSprite(RedPaddle)

    'Putting the score up
    Text.DrawText(ScoreRed.ToString(), Color.White, GameFont("ArialLarge"), 21,
511)
    Text.DrawText(ScorePurple.ToString(), Color.White, GameFont("ArialLarge"),
650, 511)
End Sub
```

Part 8 Implementing Games

At the moment the game continues even after a player scores a goal. Lets change this so that there is a short break between each goal. For this we can use the *StartUpGame* code and a new **ShowScoreScreen** procedure. The following is the new code for the *UpdateBall* procedure, you need to add the missing lines of code to your program.

```
Sub UpdateBall()
  'Tests to see if ball has hit an outside border
  'bounces the ball off the wall
  If ball.Y + ball.Height > FIELD_BOTTOM Then
    ball.Y = FIELD_BOTTOM - ball.Height
    ball.Movement.Y = -ball.Movement.Y
    Audio.PlaySoundEffect(GameSound("bounce"))
  End If

  If ball.Y < FIELD_TOP Then
    ball.Y = FIELD_TOP
    ball.Movement.Y = -ball.Movement.Y
    Audio.PlaySoundEffect(GameSound("bounce"))
  End If

  'if the ball has hit the rhs border
  If ball.X + ball.Width > FIELD_RIGHT Then
    ball.X = FIELD_RIGHT - ball.Width
    ball.Movement.X = -ball.Movement.X 'reverse the x movement
    Audio.PlaySoundEffect(GameSound("backwall"))
    ScoreRed = ScoreRed + 1

    ShowScoreScreen()
    StartUpGame()
  End If

  If ball.X < FIELD_LEFT Then
    ball.X = FIELD_LEFT
    ball.Movement.X = -ball.Movement.X
    Audio.PlaySoundEffect(GameSound("backwall"))
    ScorePurple = ScorePurple + 1

    ShowScoreScreen()
    StartUpGame()
  End If

  Graphics.MoveSprite(ball)
End Sub
```

The new ShowScoreScreen() is as follows. Add this to the top of the code file below the variable declarations.

```
  Sub ShowScoreScreen()
    DrawGame()
    Text.DrawText("SCORE !!!", Color.Black, GameFont("ArialLarge"), 220, 200)
    Core.RefreshScreen()
    Core.Sleep(2000)
  End Sub
```

Now your game should be fully functional play against a friend. Enjoy your own SwinGame!

For more tutorials and details on game development and competitions go to www.swingame.com