

## Lesson One: *Pong-style game*

### What you'll learn

- ✓ How to make objects
- ✓ How to write conditions of events
- ✓ How to set actions of events
- ✓ How to set basic movements
- ✓ Collisions
- ✓ The score object
- ✓ The lives object
- ✓ The high score object
- ✓ Backgrounds
- ✓ Active objects
- ✓ The graphics editor
- ✓ The event editor
- ✓ The frame editor
- ✓ Moving between frames
- ✓ Adding sound

### Asset List

A helpful first stage in any game's development is to create a list of needed assets. Try to come up with your own list of *sound*, *gameplay*, and *graphics* assets, then check your list against my list below.

#### **PONG ASSETS**

##### Sounds

- Ball bouncing off wall
- Ball bouncing off paddle
- Lose ball
- Background music for game (optional)

##### Graphics

- ball (must bounce off walls, and off paddle)
- paddle (player controls with mouse. Moves in a straight line).
- walls (ball bounces off: two side walls, one top wall)
- background screen
- text: title, game over

##### Gameplay

- Score (player gets one point for each time the ball hits the paddle)
- High Score board (to store player's name and high score)
- Lives (3 to begin: player loses life each time the ball is missed by paddle and exits bottom of screen)
- buttons for "start game", "quit" and "high scores"

## Pseudocode

Next, we want to write some *pseudocode*; that is, write in plain English how the game is supposed to work. This will help us plan our programming of the game. The first part of the arguments we write is called the *condition* (the “when” or “if” statements), and the second part is the *action*.

### PONG PSEUDOCODE

Frame 1: title frame with three buttons: start, quit and high scores.

If player clicks start, go to frame 2. If player clicks quit, close application. If player clicks high scores go to frame 3.

Frame 2: Ball is moving towards opposite wall at start of game.

When the ball collides with the wall, then the ball should bounce and make a sound.

When the ball collides with the paddle, the ball should bounce and make a different sound and a point should be scored.

When the ball passes by the paddle and exits the lower part of the screen, a player’s life should be lost and a “lose” sound should play. A new ball should be loaded.

When all lives are lost, screen should change to frame 3: game over screen with high scores. If player has a high score, player is asked to enter name.

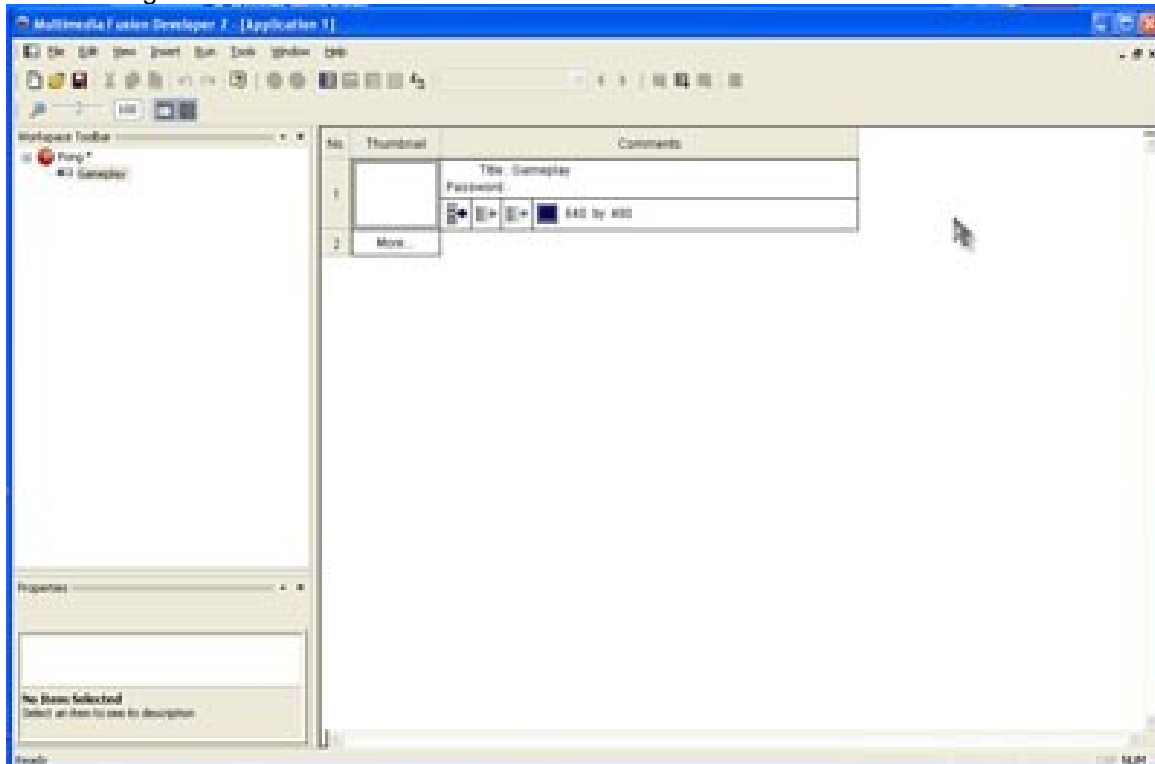
Frame 3: High score screen should have two buttons: start again, and quit game.

If player clicks start again, the game should begin again.

If player clicks quit game, the application should close.

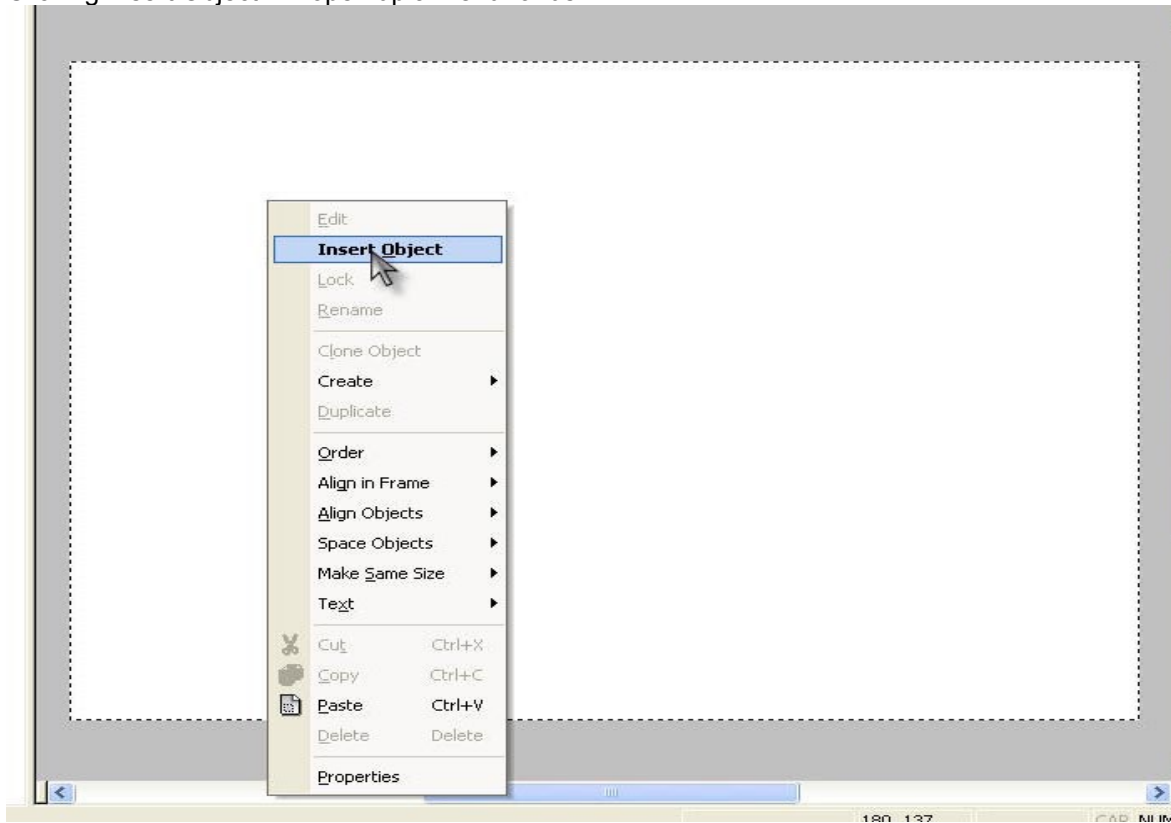
## Frame 1:Title

Okay, so let’s open up Multimedia Fusion Developer and click on File > New. Our screen should look like Image One. Right-click on the “application” name at the left and rename to “Pong”. Click on the frame title below it and change the frame name to “title”. It’s a good habit to get into to never have spaces in your asset or file names, so “title” is fine, and so is “GameTitle” or “Game\_Title”, but not “Game Title”. Once you get building more complex games or programming, this will all make more sense, but for now, let’s build some good habits!

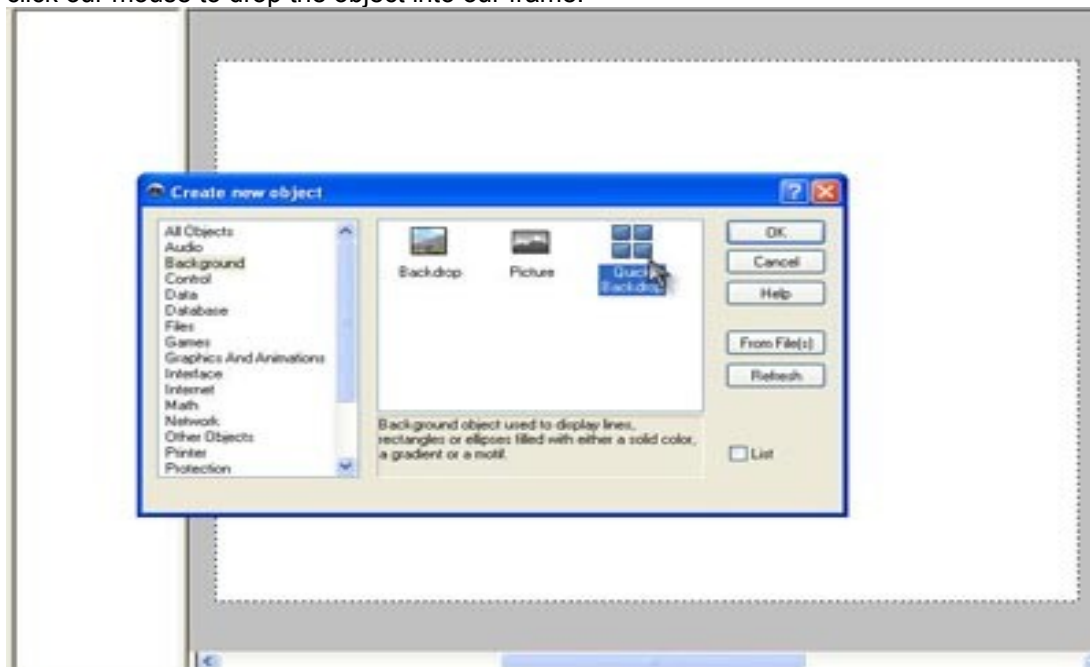


Double-click on the white box below the word “thumbnail”. Your title frame will now open up.

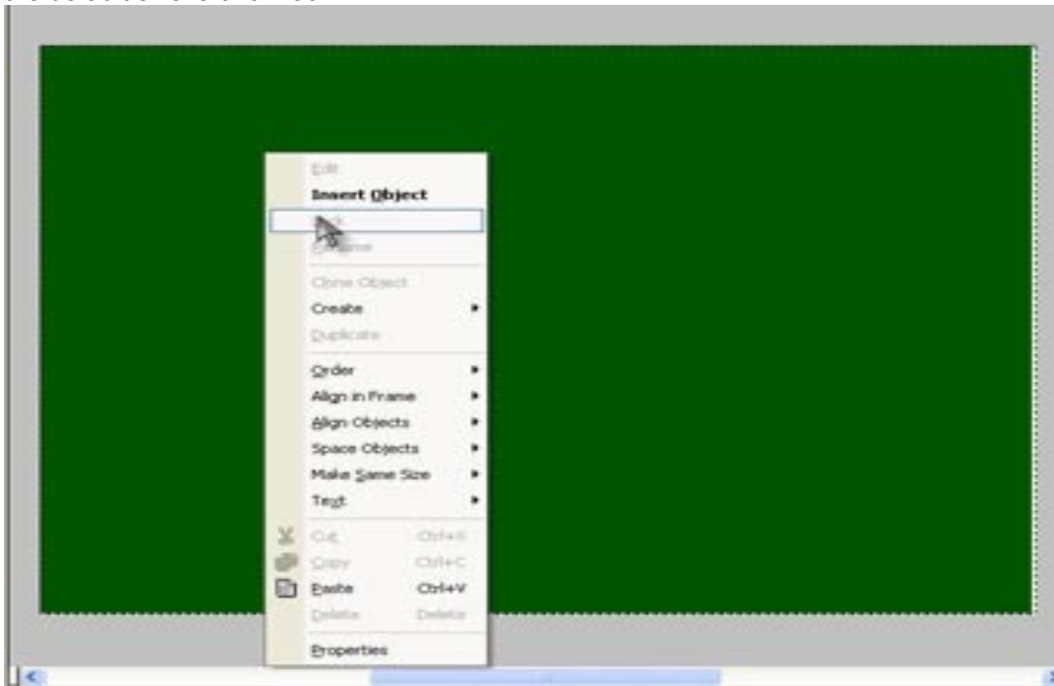
Right-click on the large white frame and click on Insert Object. An “Object” is any asset in the game. Clicking Insert Object will open up a menu for us.



First, we'll make the background, so click on the BACKGROUND menu. There are three options here. We could import our own graphic, make our own graphic in the software, or take the simple “quick” route and have a solid (or gradient) colour. We'll do that for now. Select “Quick Backdrop”. Nothing happens until we click our mouse to drop the object into our frame.



Now we have just a boring grey square. We need to click on that until we're given some re-sizing beziers on the sides of our square. Resize the square to fit our background, then look on the left to the "Properties" bar that's opened up. We can change the colour/shading here. We can also set the size here: the default size is 640x480.



In the library, rename your backdrop object "background".

Now let's put out Pong text on there. Right click our background. It's often easiest if you select "lock" when you right-click a background, to lock it in place so it doesn't move later. Select Insert Object> Text> Formatted text.

This will bring up a textbox. Type in "Pong Game". You can adjust the font by the "text" in our toolbar at the top of the software. Under "properties" in the bottom left corner, deselect the auto-vertical scrollbar and resize your text frame to the size of the text. Name your text "pong\_game" in the library toolbar.

Now for the buttons. Rightclick on the background, and choose Insert Object> Interface Object> Button. Do this three times.

1. For the first button, under properties change the settings for text to "Start". Rename this button start in the library.
2. For the second button, under properties change the settings for text to "Quit". Rename this button quit in the library.
3. For the third button, under properties change the settings for text to "High Scores". Rename this button highscores in the library.

Let's take a look at our pseudocode;

"Frame 1: title frame with three buttons: start, quit and high scores.

If player clicks start, go to frame 2. If player clicks quit, close application. If player clicks high scores go to frame 3."

So we need to set some *events--some things to happen-* in the Event Editor. Click on the icon above the frame space that looks like a grid, called Event Editor (or hit CTRL>E). This brings up the event editor, where we enter our conditions, and our actions. The left side holds the conditions, and the right sold holds the actions.

Our condition for each button will be "when the user clicks the button"

So on the left hand side, under #1, right click the start button and select "Button clicked?" This will write the code, "button [button] clicked?" as our condition. Do this for each button.

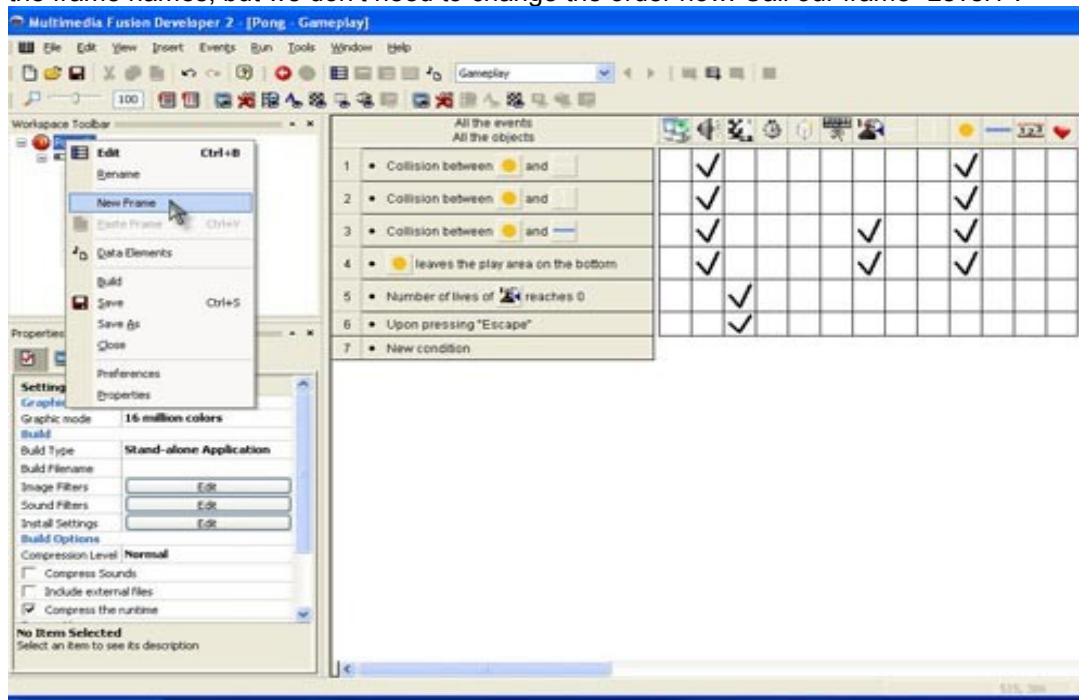
On the right hand side are our actions. All three of the actions for these buttons relate to the storyboard-- we want to go somewhere in the game. Under the box for the storyboard icon, right-click and select for line #1, "Next Frame". You'll see a checkmark appear in the box.

For line #2 (quit), select "end the application"

For line #3 (highscores), select "jump to frame". We haven't made any other frames yet, so we'll have to leave it for now and come back here later.

## Frame 2: Level1

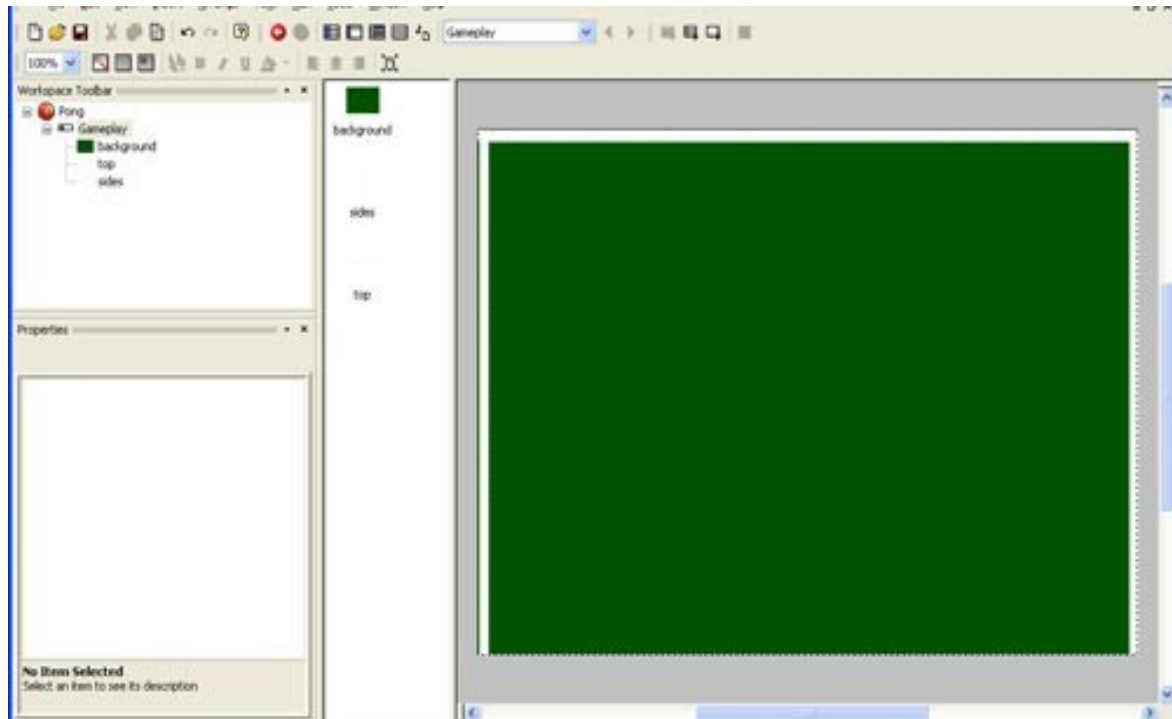
Click on our game name "Pong" in the Workspace Toolbar and select "New Frame". You'll see it automatically places the frame after the first frame. We could change the order by dragging and dropping the frame names, but we don't need to change the order now. Call our frame "Level1".



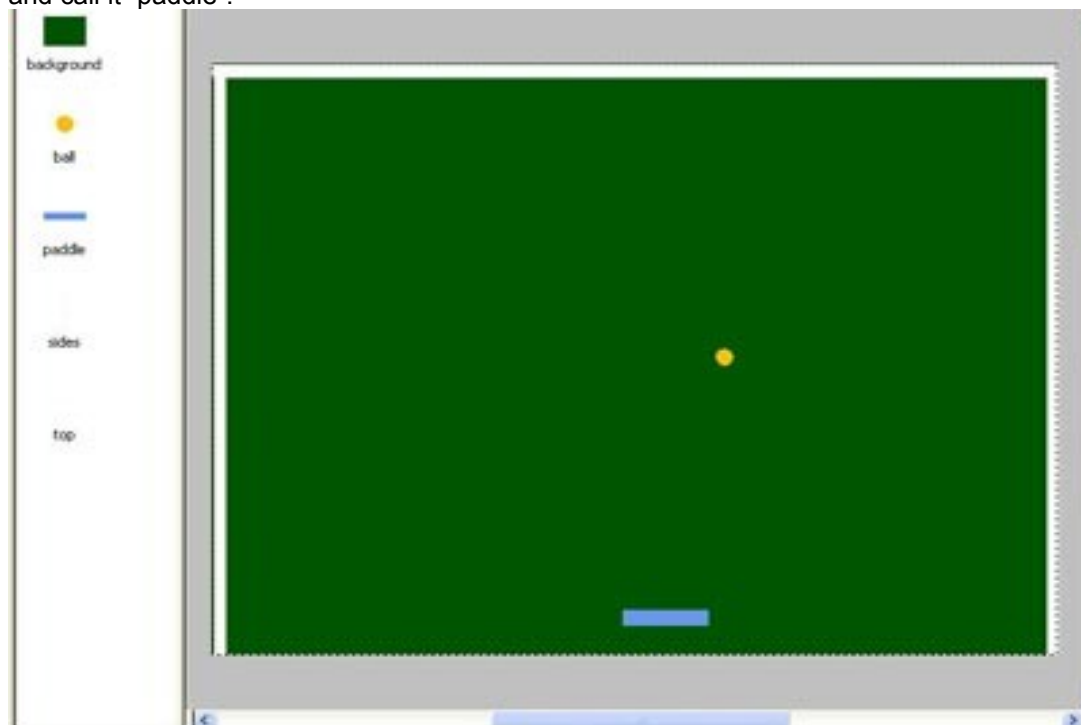
Create a background just like we did in frame 1.

Okay, now we need to build some walls. Because we're interacting with the walls in the game (we're bouncing a ball off them), we need to make the walls an ACTIVE OBJECT. Right click on the backdrop, and while we're here, let's click "lock". That way the backdrop won't be moving around as we adjust objects on it. Right click again and click on insert object. This time choose the "Graphics and Animations" menu and select "Active". Drop the object into our frame and you'll see we have a blue diamond shape. Let's change that into a wall. Double click the object to bring up the graphic editor.

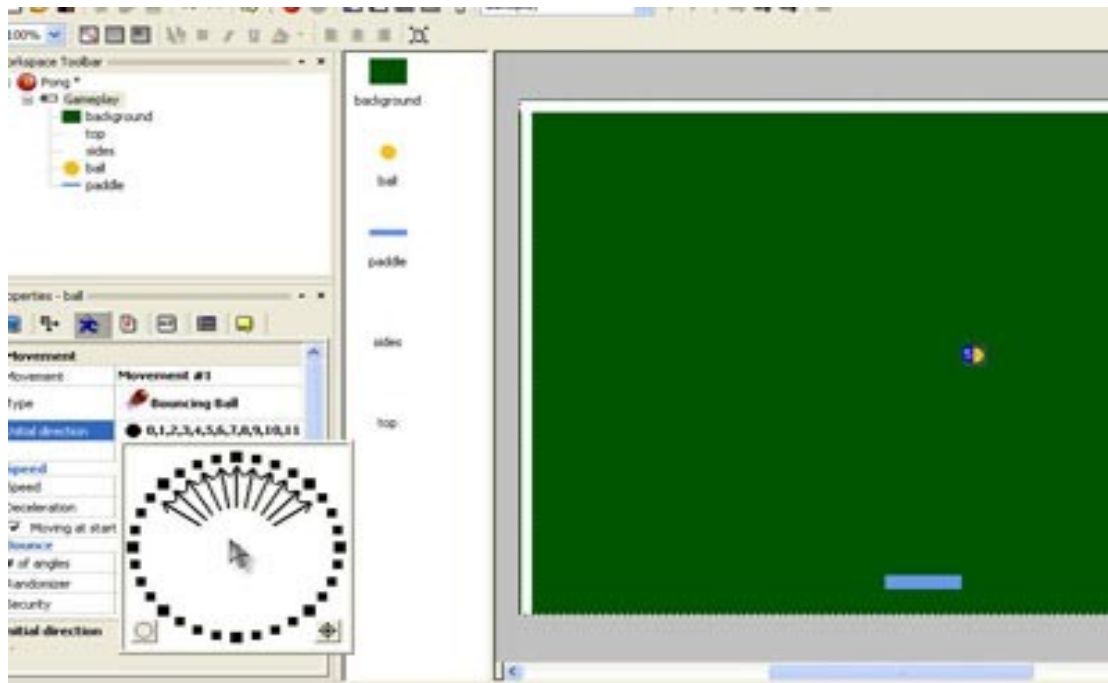
Select the select-rectangle shape and delete the diamond. Change the foreground and background to white, and select the "fill color" option. Now draw a white rectangle. Click ok. We can resize the rectangle to fit the frame and make a side wall. Copy and paste that wall again for the other side, then make a new object for the top wall. You will see our objects are now in the "library" of the game. We should give them appropriate names. Change the backdrop to "background", and change the two side walls to "wall" and the top wall to "top". Your game should now look like this:



Now we need to create a ball and a paddle. Can you guess what kinds of objects they will need to be? That's right, ACTIVE objects again. Make a round yellow ball and call it "ball", and a rectangular paddle and call it "paddle".

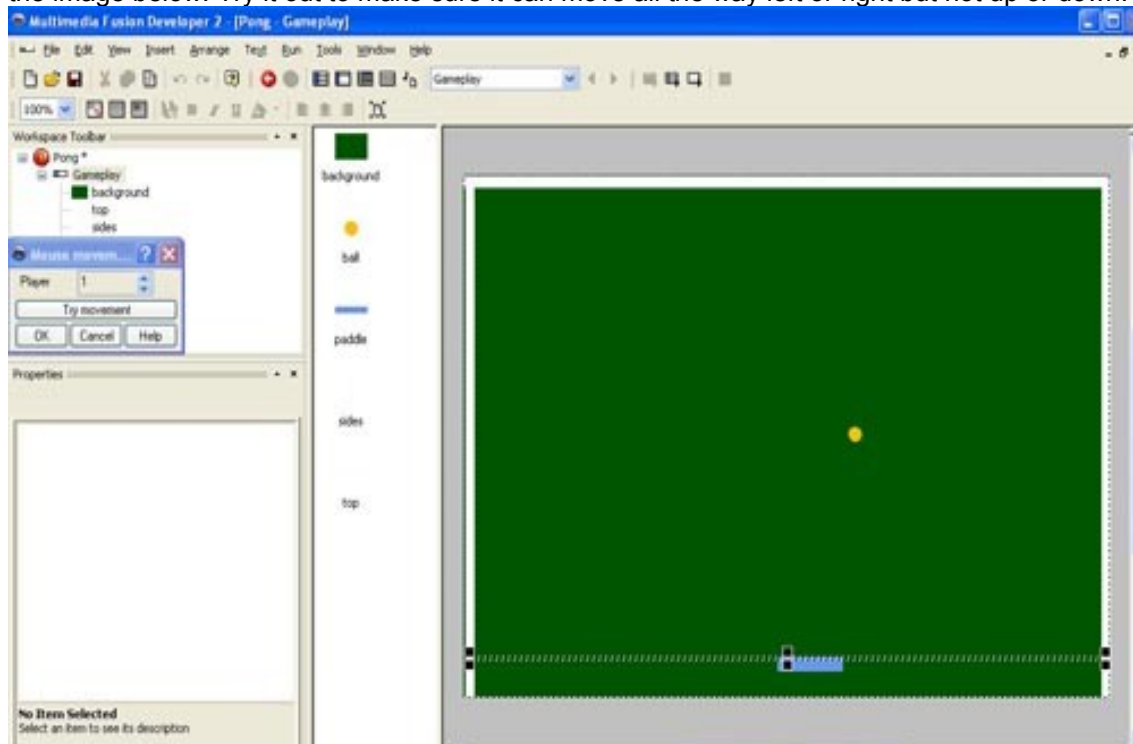


Now let's add some movement options to these two assets. Click on the ball and you'll see it opens up a properties window on the left. Click the little running man to open the movements section. Here you'll see it has defaulted to "static". Click that and change it to, you guessed it, "bouncing ball". Now we can adjust the traits of the bounce. We want to adjust the direction so that the initial direction is facing the top wall. If we click beside "initial direction", we're given a circle of options. Click the icon on the bottom left to clear all the arrows, and then select a few to point to the top.



You can play around with the other options later: random adjusts the randomness of the bounce, speed the speed, and so on. Check the box for “moving at start” and click on the “try movement” if you want to see it in action. Escape out of that screen and then let’s add some movement to our paddle.

Click on the paddle to open the properties window. This time we want to have the paddle mouse-controlled by the player. This will give us a box near our paddle. Reshape it until it’s a thin long line like the image below. Try it out to make sure it can move all the way left or right but not up or down.



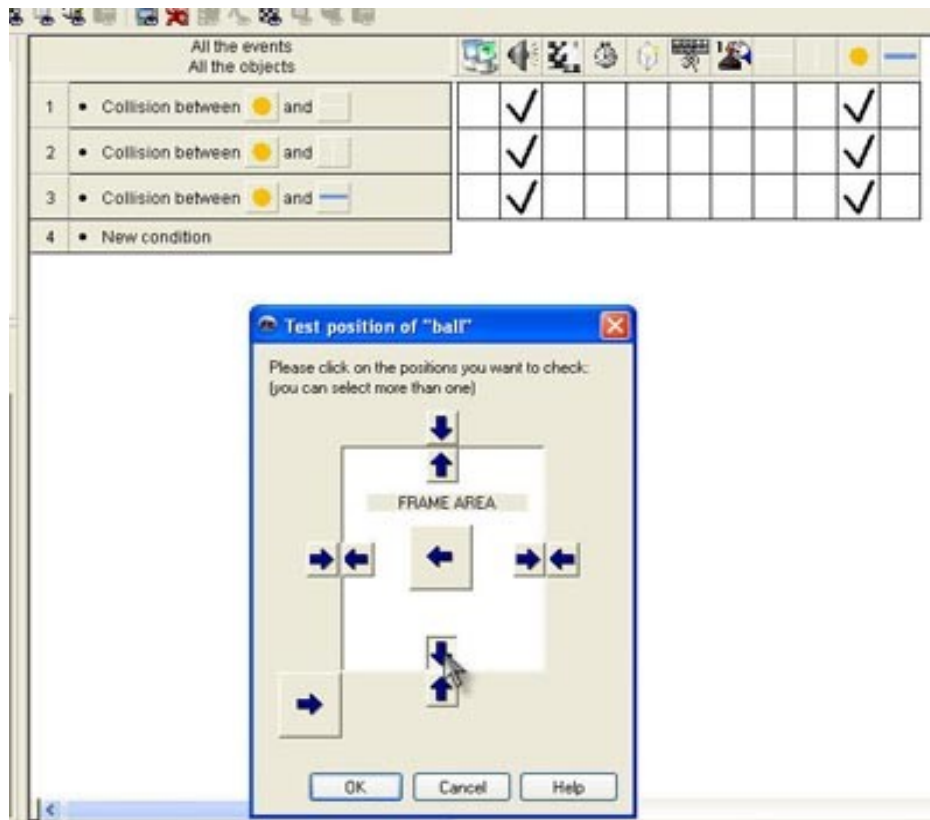
Now we need to add some actions--some EVENTS--to the game. It's a good time to look at our pseudocode again. The first conditions we want to add are our "when" conditions we wrote before:

- When the ball collides with the wall, then the ball should bounce and make a sound.
- When the ball collides with the paddle, the ball should bounce and make a different sound and a point should be scored.
- When the ball passes by the paddle and exits the lower part of the screen, a player's life should be lost and a "lose" sound should play. A new ball should be loaded.

Okay, right-click where it says "New condition" on line one. It will bring up a menu of game objects. Right-click on the ball and select "collides > another object" Now we will select the "wall" object.

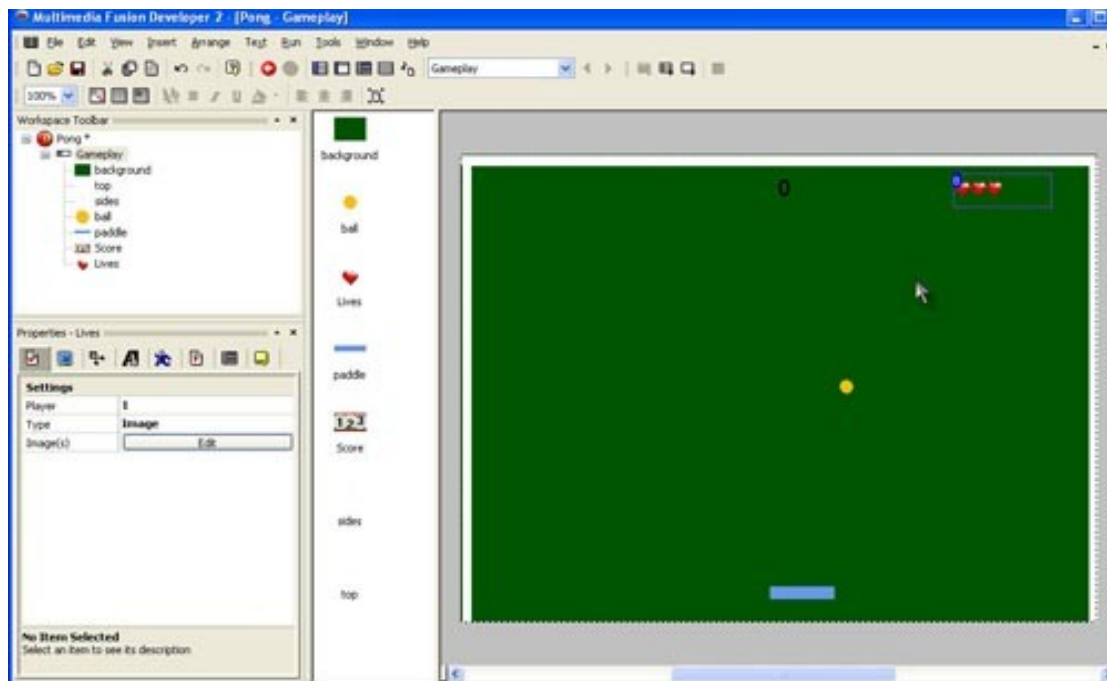
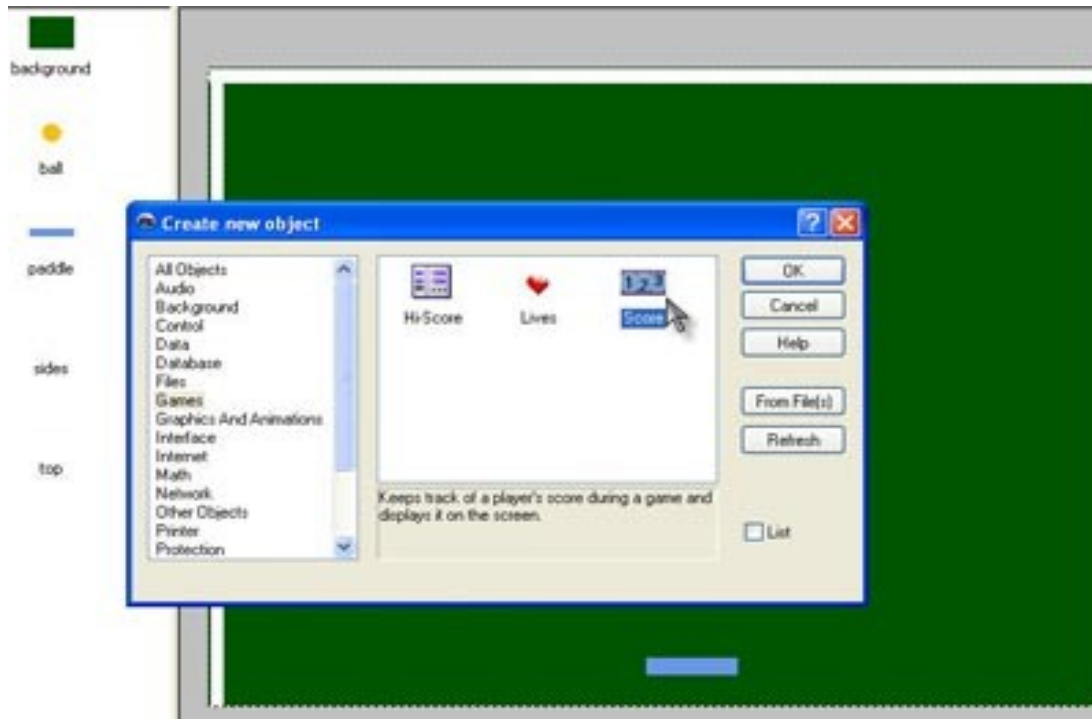
Do this again for the top wall, and again for the paddle.

Now right-click the ball again and select "Position > test position" This will bring up a frame. Select the arrow pointing out of the bottom of our screen.

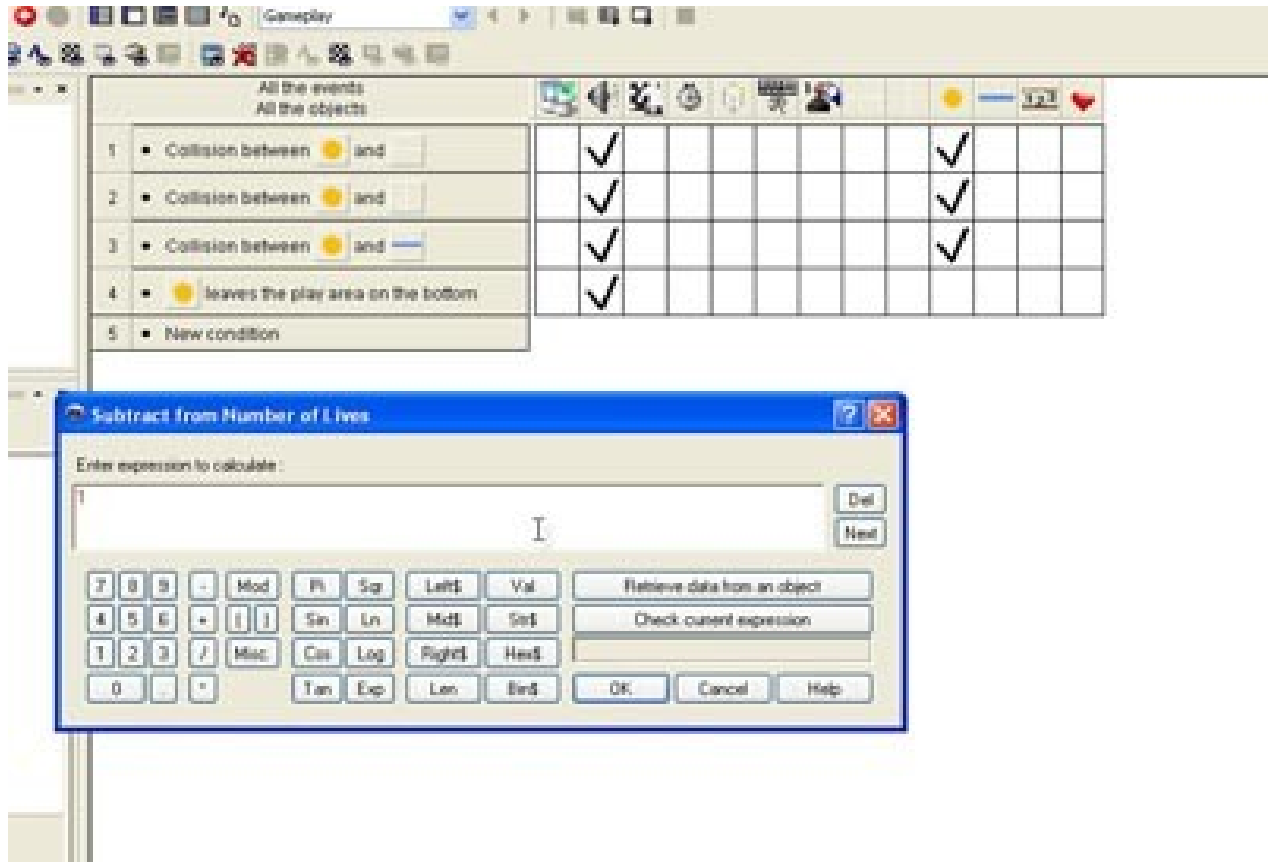


Now let's go in and add some *actions* as I have above. Okay, so the first line is our "when the ball collides with the wall"... and our actions: the ball should bounce off the wall, and it should make a sound. You can see I've clicked underneath our ball icon on the action side of the event editor, and selected "Movement > Bounce". I can then drag that box to the two lines below as well, to save time. Now let's make a sound. Right-click on the box under the sound icon, and click "Sample > Play sample". This will allow us to browse our files and find the sound "bounce1.wav". Do the same for the next wall. For the paddle, play "bounce2.wav". You'll find the sound files already prepared for you in ZIP a file on my website, [www.gamessound.com](http://www.gamessound.com). These sound files were downloaded from [Freesound.com](http://Freesound.com). The MIDI file I used is from the NES game Rad Racer, and was downloaded from [VGMusic.com](http://VGMusic.com).

Our fourth line needs us to go back to our frame screen and add a few components-- the scoreboard, and the lives icons. First, let's get a new ball loaded once the ball exits the screen. Click under the ball action and select Position>Select Position. This will give us some cross-hairs to position the new ball. We want to make sure it's heading away from us at the start, so click on Direction and select appropriate arrows. Double-click on the left-hand side of the screen where we have our Gameplay frame listed. This should take us back to our frame screen. Now Insert Object once again, and this time pick "Games > Score", drop it on the screen, and then do the same for Lives.



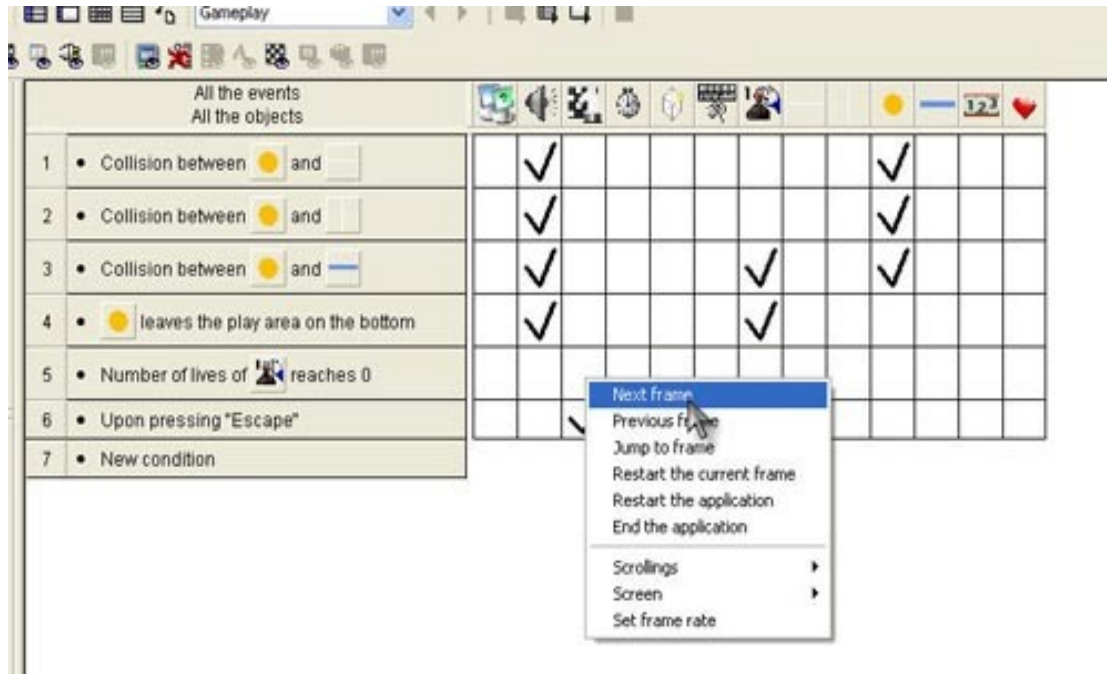
Now let's go back to our Event Editor and set some more actions. First, we want to score a point any time the player hits the ball with the paddle. Under line 3, "Collision between ball and paddle", let's click under the hand with a joystick icon, and select "Score > Add to Score". This will bring up an expression editor. We just want to add 1 to the score for every time this occurs, so change the 0 to a 1 and hit OK.



Now let's subtract a life when the player misses the ball. On line 4, click on the box under the joystick icon again and select "Number of Lives > Subtract from Number of Lives". Change the 0 to a 1 and hit OK. Going back to our Pseudocode, there are still a few things to do:

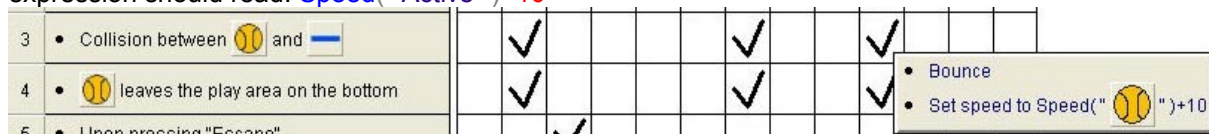
- When all lives are lost, screen should change to game over screen with high scores. If player has a high score, player is asked to enter name.
- High score screen should have two buttons: start again, and quit game.
- If player clicks start again, the game should begin again.
- If player clicks quit game, the application should close.

So let's add a new condition on line 5. Right-click and select the game icon (joystick and hand). Right click "Number of lives reaches 0". Now we want to add an action under our storyboard (which organizes our frames) and select (go to) "next frame". We haven't made the next frame yet, but we'll get there.



One more thing to do while we're here--we'll give the player the option of quitting early by pressing "escape". Click on new condition on line 6 and right-click the keyboard-like icon. Select "Keyboard". It will ask us what key we want to set an action to. Hit your ESC key. Now let's add under the STORYBOARD icon that (if the player hits the ESC key...) we want to End the Application.

We also want to increase the speed every time the ball collides with our paddle. So under the condition "collision between ball and paddle" select under the ball, and add the following action: Movement> Set Speed. This will call up our expression editor. We want to find out the current speed, and then add 10, so retrieve data from current object> ball>speed will find out what our current speed is, then add +10, so our expression should read: `Speed("Active")+10`



### Frame 3: Highscores

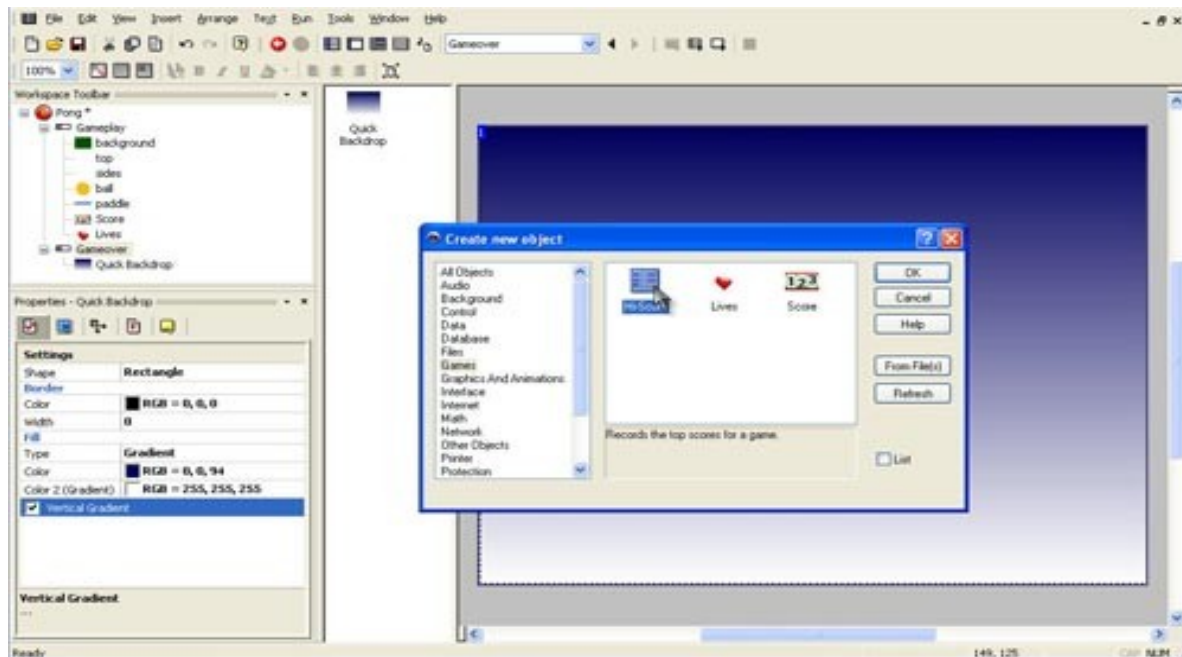
Let's go back and add that final game-over frame now (When all lives are lost, screen should change to game over screen with high scores). Insert a new frame.

Rename our frame "highscores".

Click on our frame's thumbnail and change the background to a quick backdrop.

Let's add some text that says "Game Over". Right-click on the stage and hit "Insert Object > Text > Formatted Text". This will call up a text box that we can click into and type Game Over. By selecting the "text" menu at the top of our screen, we can adjust the font's colour, size, etc.

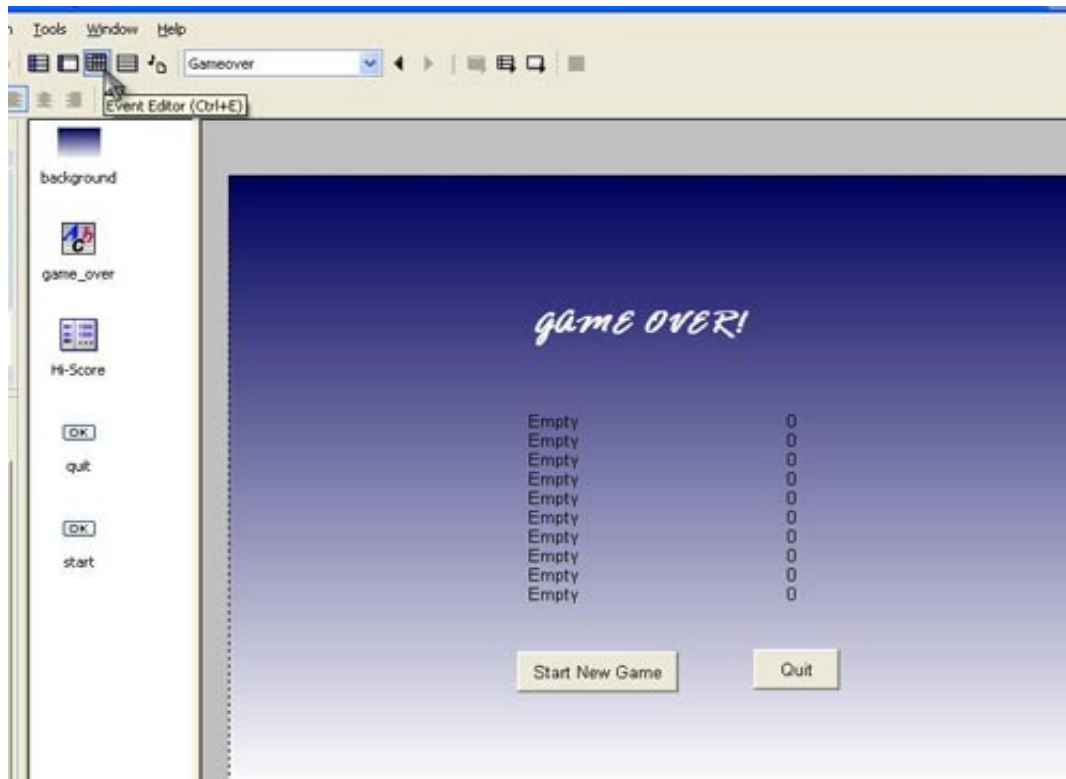
Now drop in a High Score table. Insert Object > Games > High Score. It will come up with "empty" and zeros, but that's because nobody has saved their scores yet!



Now we're almost done:

- High score screen should have two buttons: start again, and quit game.

Let's drop some buttons on the stage. Insert Object > Interface > Button. In the properties, change the type on the first button to "Start Game". Rename the button to "start". Drop another button on, changing the type to "Quit" and naming it "quit".



Finally, we need to add some actions so that something happens when the buttons are clicked.

- If player clicks start again, the game should begin again.
- If player clicks quit game, the application should close.

So open our event editor again, select the first line (since we're on a new frame, the events screen for this frame is blank. Click on line 1, right-click and select our button "start". Right click that and select Button is clicked. Do the same for the quit button.



Now to set the actions: Under the Storyboard icon, change the option to "Restart Application" for "start game". For quit--you guessed it, "end the application".

Now go back to Frame one and fix the event for the highscores button so that it jumps to frame> frame 3.

Now we've created the basic Pong Game. You can hit F8 to try out the game!

Don't forget to save your project.

### Additional Exercises

Too easy for you? Try these additional exercises:

1. Add a title screen with buttons to START, QUIT and HIGH SCORES.
2. Change the control from mouse to keyboard with two direction options (left and right), controllable by arrow keys (hint: use Eight Directions Movement).
3. Change the score from an increment of 1 to 10 (hint: EDIT, don't ADD the increment!).
4. Start some background MIDI music when the gameplay frame loads (hint: use the storyboard icon to have the music start upon the loading of the frame). Play with the settings so that the music stops when the player loses a life, and starts up again when a new ball is loaded.
5. (bonus). Add a second level after a player reaches a score of 100 (hint: compare player's score equal to 100) Cut-and-paste the gameplay frame so you don't have to repeat yourself. Rename the level 2 frame to "gameplay2". Don't forget to edit any "next frame" actions in the Gameplay event editor appropriately.