

Lesson Three: *Arkanoid-style game, developed from Pong-style game*

What you'll learn

- ✓ Global values (variables)
- ✓ Basic animations
- ✓ Changing the state of an object
- ✓ Group Events (functions)
- ✓ Randomly changing location of an object by swapping with another object
- ✓ Creating objects in real-time in the game
- ✓ Motion paths
- ✓ Shooting
- ✓ Destroying objects
- ✓ Inserting a second conditional statement (so condition a AND condition b)

Asset List

Google Arkanoid and play a game online. Then create your own asset lists and some pseudocode on your own, and check it with my list below. For *Arkanoid*, we want to add some bricks that the player should have to destroy with the ball. We'll make three additional levels. We'll keep the "Gameplay" frame from our Pong game as level one, "Gameplay2" as level two, then "Gameplay3" as level three, and so on. Level two will just have standard bricks that have to be hit twice to be eliminated. They should change colour the first time they are hit. Level three will have a bonus brick hidden somewhere in the field of bricks: a "shoot" bonus where the player will temporarily have bullets to shoot at bricks.

ASSETS

Sound

As before, *plus*:

New sound: brick hit sound

New sound: brick explosion sound

New sound: "win item" cue for bonus bricks

New sound: fire bullet

Graphics

As before, *plus*:

- bonus pill fall from top when bonus brick is hit
- bullets

Animations:

- paddle (player controls with arrow keys. Moves in a straight line). Paddle must be able to change to “gun shooter” paddle and “go large” paddle.
- bricks (must change colour upon being hit the first time).
- bonus bricks (should look like bricks) randomly hidden amongst regular bricks

Gameplay: As before. Plus: space bar shoots bullets.

PSEUDOCODE

Level one see Pong.

Level two: as with level one, but with additions:

- when ball collides with brick for the first time, brick should change colour and make a sound.
- when ball collides with brick for the second time, brick should be destroyed and make a sound
- when a brick is destroyed, players score should increase by 100 points
- when all bricks are destroyed, go to next level

Level three: as with level two, but with additions:

- two bonus shootbricks should be randomly hidden amongst bricks.
- when ball collides for second time with shootbrick, bonus pill should fall from the top of the screen.
- If player collides with pill, shooter animation for paddle should load and player should be given 10 bullets to shoot at bricks.
- when space bar is pressed, bullet should fire and make a sound.
- When bullets reach 0, return to regular paddle.
- When bullets hit brick, brick should explode on second hit, as with ball collision.
- If player misses pill, nothing happens.

Frame 1: Title

Open up our (Lesson 1) *Pong* game.

Frame 1 will stay the same.

Once you've successfully solved the additional exercises for Lesson One and worked your way through Lesson Two, let's make this game more complicated! We're going to use the original one-player *Pong* file we've already created, but let's save it as a new application in case we make mistakes. Call this application *Arkanoid*.

Edit your title frame so that it now reads "Arkanoid" instead of "Pong Game."

Frame 2: Level1

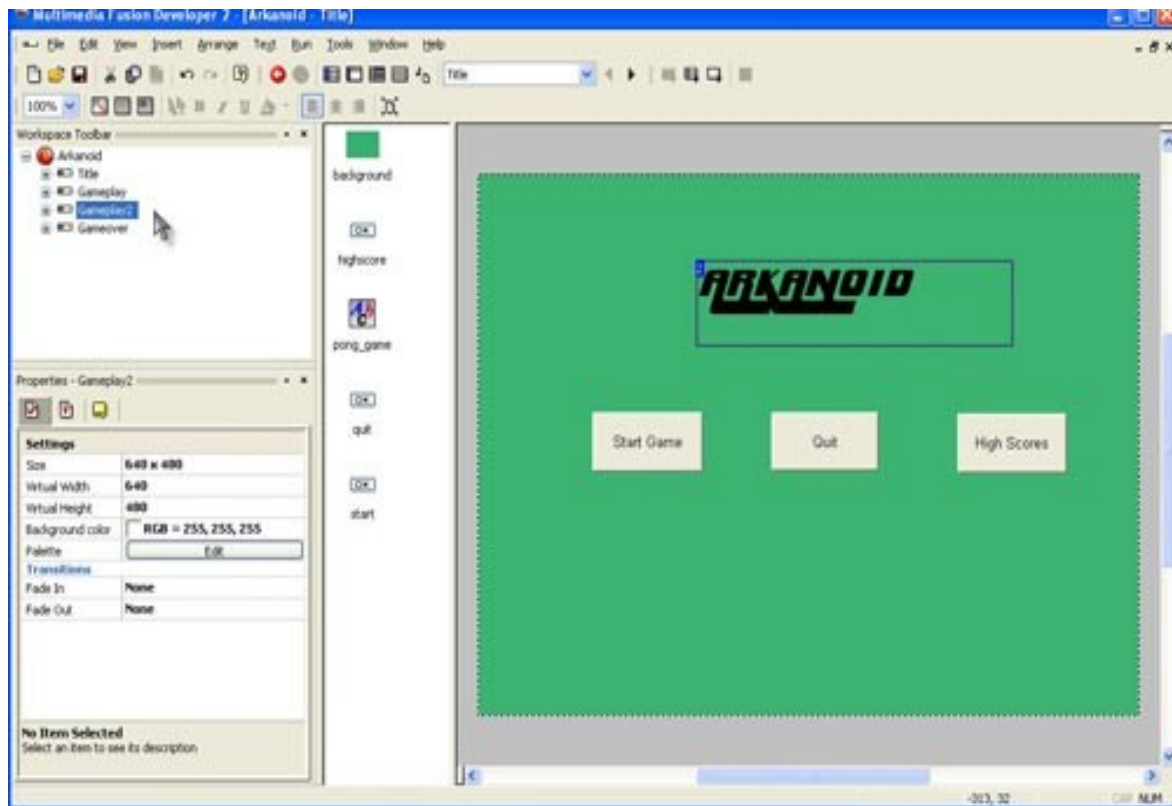
Use Level1 from Pong game.

Go into the Event Editor and add the following condition:

Player1 > Compare to Player's Score > 10. We've just set a condition that says, "when the player's score reaches 10". Set the action to Storyboard > Next Frame.

Frame 2: Level2

In the Workspace Toolbar, clone level1 to make a new frame. Name it level2. Move it above the highscore frame in our frame list.



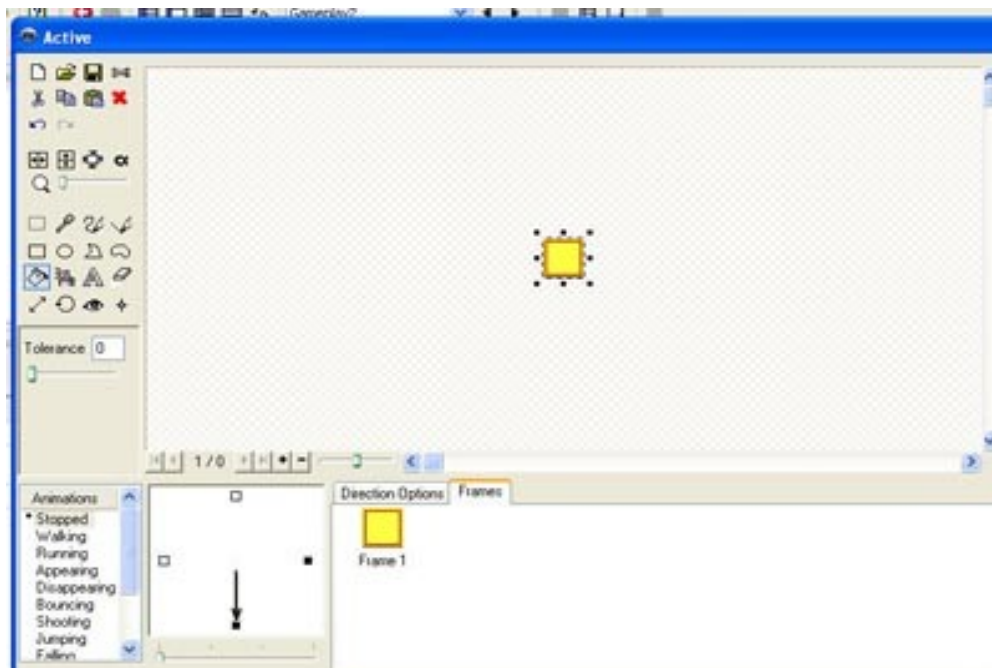
First we need to check our existing code. On the title frame, “quit” should now take us to the last frame, frame 4. On our level1 frame, when number of lives = 0, should read go to frame 4. When score = 10, go to level2 frame. These should have changed automatically, but check them anyway.

Now for the level2 frame. Open the event editor: delete the line that says “when score = 10, go to next frame” because we’ve already reached 10. Right now, the score object won’t carry forward since we haven’t set global parameters. For now, add an event that reads:

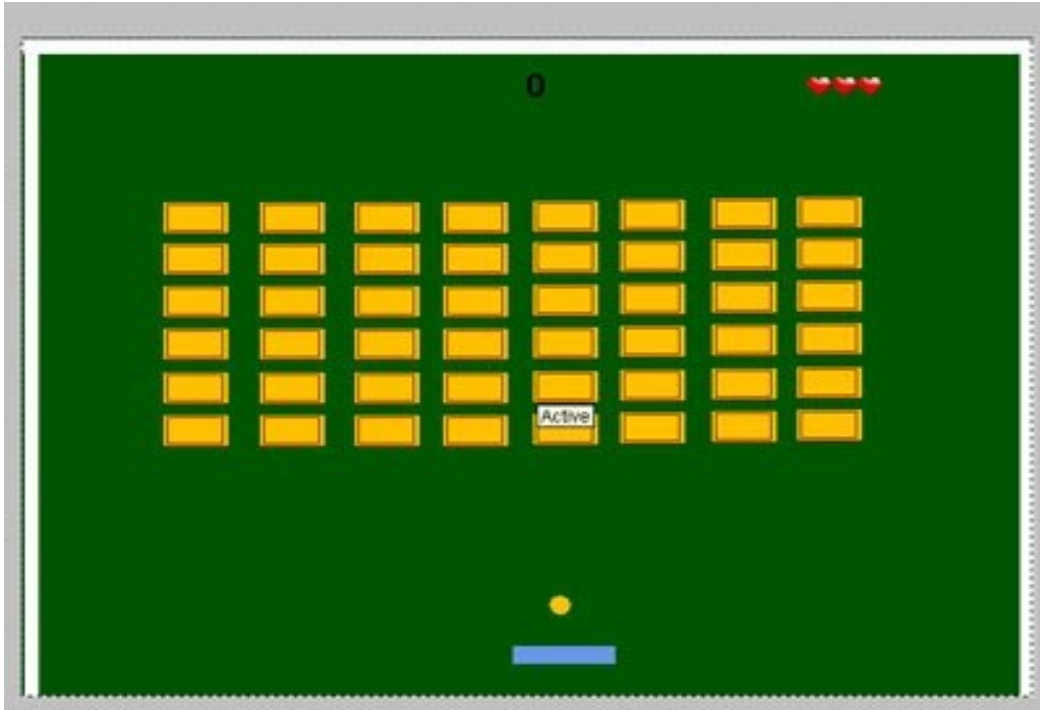
Storyboard>Start of Frame and under actions, Player1> Set Score> 10.

Open the Frame Editor.

Drag the ball to the lower part of the screen to get it out of the way and drop a new Active Object on the frame stage. Change the graphic to a rectangle resembling a brick. Still in the graphic editor, we need to alter the brick so it has a second stage for after it’s been hit once. Select and Copy your brick. Where the arrow points to the right, beside the animation list, we’re going to select the “down” arrow. Paste your brick back in here and adjust the colour brightness by filling the brick with a brighter colour. We now have two brick states depending on “direction” options: one regular, one bright. Hit OK.



Rename our Active object to “brick”. Now we need to copy and paste our brick until we have a large number of bricks on our screen: ten across, five deep. The easiest way to do this is to use the “duplicate” action. Right-click our brick and select duplicate. Insert 10 columns and 5 rows. Have a space of 2 between rows and columns.



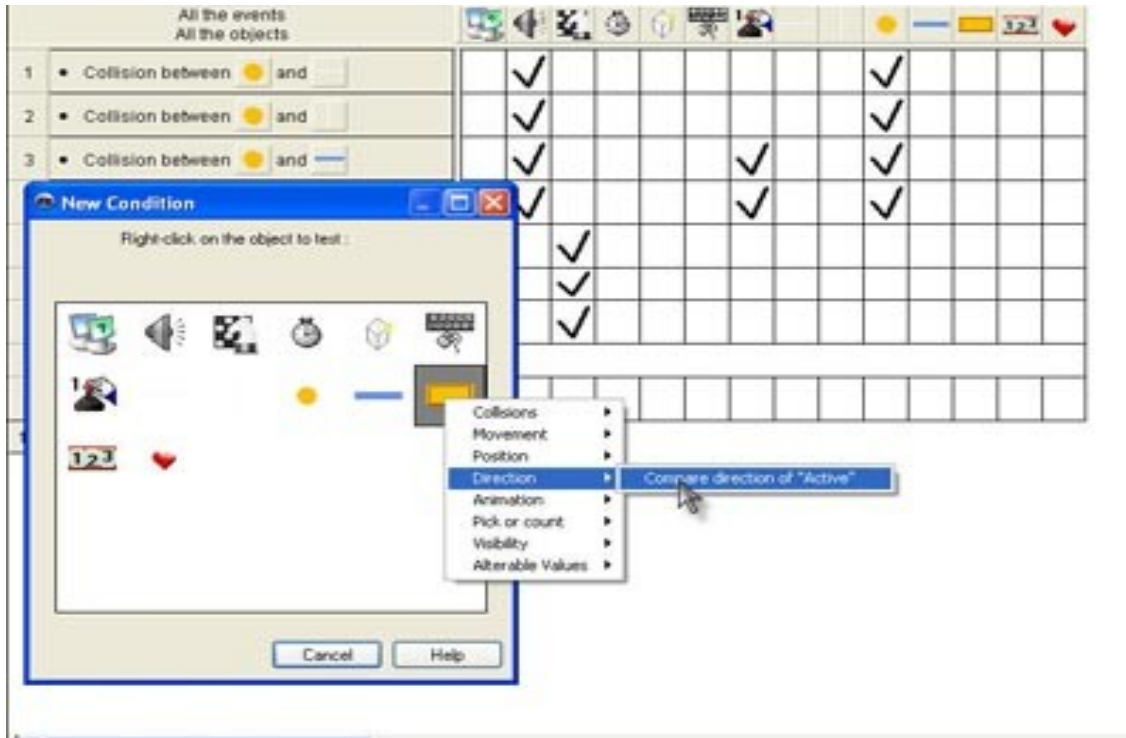
Okay, let's go in to our Event Editor and set some brick events.

First, let's add a comment to our event editor so we can separate what's special to this particular level. Right-Click on the number-box for line 8. Insert > A comment. Type "level two". Now we can separate our code out more easily.

All the events		All the objects											
1	Collision between and	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	Collision between and	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	Collision between and	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4	leaves the play area on the bottom	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5	Number of lives of reaches 0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
6	Upon pressing "Escape"	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
7	Score of = 100	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
8	Level Two												
9	New condition												

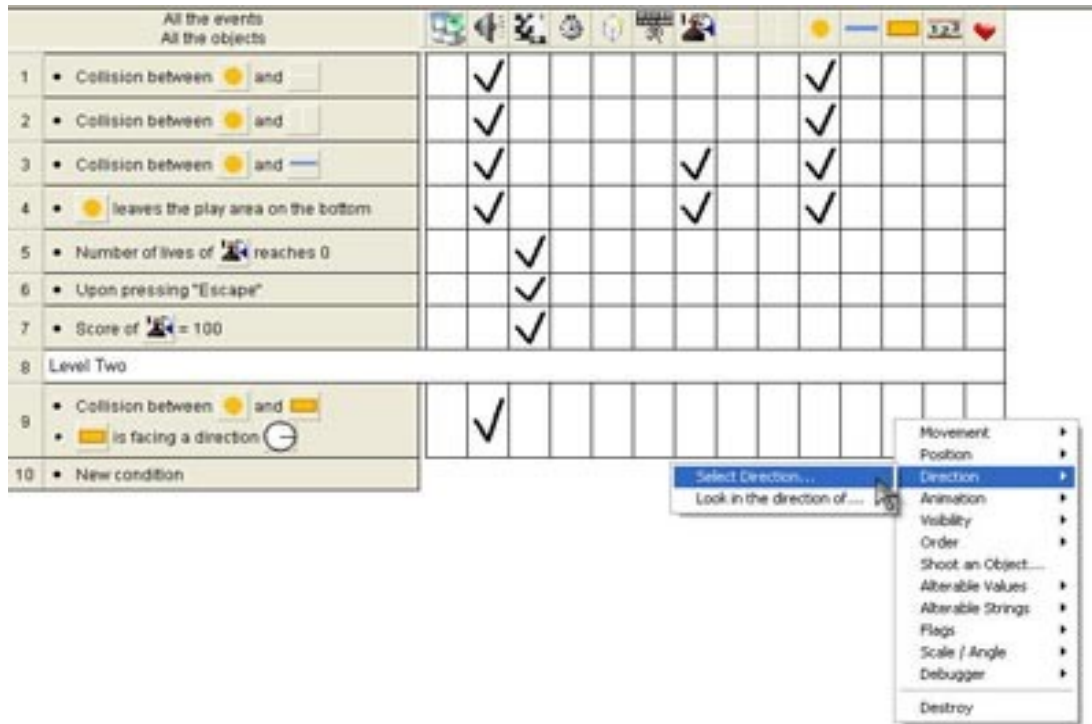
Check our pseudocode: “when ball collides with brick for the first time, ball should bounce off brick, brick should change colour and make a sound.”

So you should get the first part: when ball collides with brick. So how do we specify “for the first time”? Remember those directional arrows we set? Right click on our conditional statement and click “insert.” This adds ANOTHER condition to this condition: So think of it as “when ball collides with brick AND ...” right-click brick, Direction > Compare Direction of brick to select brick facing right hand side.



Now for our actions: “ball should bounce”, “brick should change colour” and “make a sound”
“Bouncing” the ball and inserting a sound should be easy for you now. Use “impact.wav”.

Now to change colour. Under the “brick” box, select Direction > Select Direction and change it to “down”.



Try that out by pressing F8. You'll have to get your score to 10 before you can test your level. Or you can go back to the original frame, and adjust the start game button to jump to this frame while we cheat!

Now for our second brick action: "when ball collides with brick for the second time, brick should be destroyed and make a sound"

Once again, When ball collides with brick is our first condition. And we now need to add a second condition, as before: Insert > brick > compare direction of : this time, we want to compare it to the down arrow. Let's have this action play the sample "plode.wav". now we need to destroy the brick. Under brick, right-click the box and select "Destroy" (the bottom of the list).

All the events All the objects													
1	Collision between and	✓								✓			
2	Collision between and	✓								✓			
3	Collision between and	✓					✓			✓			
4	leaves the play area on the bottom	✓					✓			✓			
5	Number of lives of reaches 0			✓									
6	Upon pressing "Escape"			✓									
7 Level Two													
8	Collision between and	✓								✓		✓	
	is facing a direction												
9	Collision between and												
	is facing a direction												

One thing we need to do is change the order of these two conditions: Because it should check the second condition (is the direction facing “down”) before the first, and then it will see, “no the direction is not down” and then move on to the second condition. Just drag and drop those.

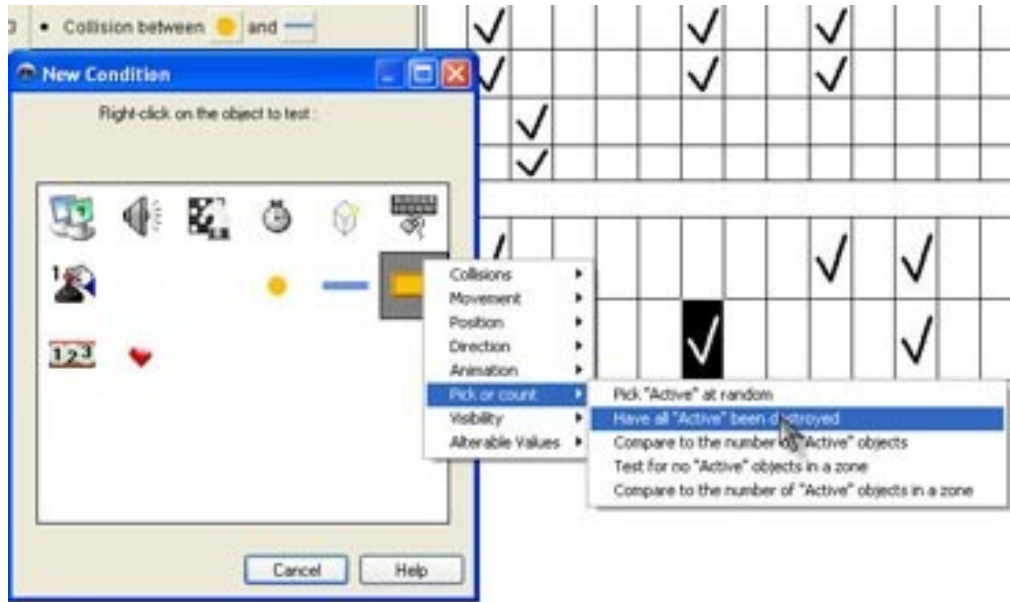
Check our pseudocode again: “when a brick is destroyed, players score should increase by 10 points”
Okay, so lets do that. (Player icon> add to score> 10).

The screenshot shows the same event editor as above, but with a context menu open over the 'Add to Score' option. The menu items are: Score, Set Score, Number of Lives, Add to Score, Player Control, and Subtract from score. The 'Add to Score' option is highlighted.

“when all bricks are destroyed, go to next level”

That should be easy enough. New condition: Brick> pick or count > have all “bricks” been destroyed?

Then our action is to go to the next frame. Try it out, save our file, and we'll be back for Level Three next!



Frame 4: Level Three

Clone our level2 and rename the new frame level3.

We cheated in the last level when we set the score, because we knew the score would be 10 (when player reached 10 on level 1, then level2 would load). Normally, we need to set special VALUES if we want to carry over information from a previous frame to the next frame.

We want to be able to retrieve the current score to set it at the start of this level and set it as our new score. In the event editor for level 2 (frame 3), at the top under our “start of frame” condition, set a new condition that reads “end of frame” (storyboard> end of frame). Now under “special” (looks like 2 computers), set global value> retrieve value from the player1’s score. This says, store a global value (global meaning stored across the whole game, rather than just one frame), and that global value is the current score of player 1. It defaults to the variable A. That’s fine for now.

In Level 3’s event editor now (frame 4), let’s retrieve that value at the start of the frame. Delete the line that says “start of frame> set score to 10”, and re-write it as, “at the start of the frame, set score to global value A”. This will retrieve the value of A and set that as our score!

Now we need to create a bonus object that drops from the top of our frame when we destroy a bonus brick. We also need to create a bonus brick that is randomly hidden amongst our brick objects.

First, copy and paste our brick object so that it creates a new object. Call that object bonusbrick.

Replace two regular bricks with three bonusbricks on the screen.

Now to randomize those bricks!

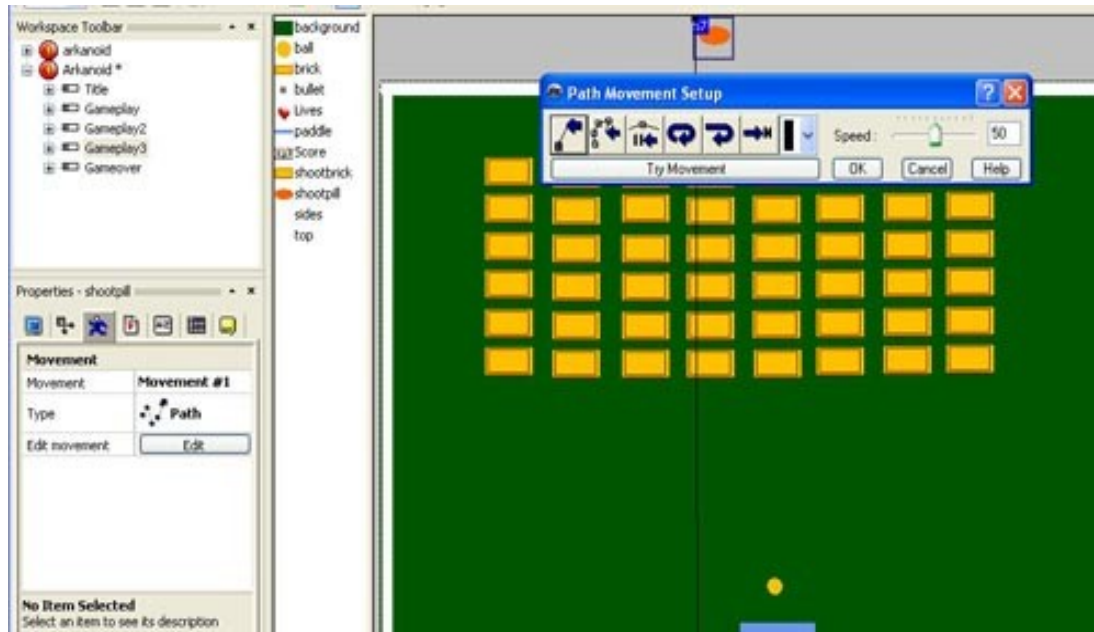
Create a new condition that reads (under “special”), limit condition> repeat 10 times. Under actions, select under the first brick and choose Position> Swap position with another object. Drag and drop that screen to the other brick object. Now our bricks will be randomly swapped!

Now create a bonus object, and call it “bonus” and put it off the top of the frame.

Copy our Level2 code, only change the “brick” object to “bonusbrick”, so that we have a set of events that reads, “when the ball collides with bonusbrick for the first time, change direction, play sound and add 50 to score”, and, “when the ball collides with a bonusbrick for the second time, destroy the brick, play a new sound (win) add 50 to the score” and then do the following;

We want to CREATE an object: this is the weird diamond-looking icon. Create the “bonus” object just off the top of the screen when we hit the bonusbrick for the second time. So on second hit, create object>bonus (and set position off-screen).

Go back to the frame editor, select out bonus object, and give it a motion > path pick the first icon and draw the pathline straight down through the bottom of the frame.



Now back to the event editor. Select under the bonus object for our second collision, and choose “Motion> Start”.

We’re going to set a Group of Events for what happens when we collide with (collect) the bonus. A group of events is a special series of grouped actions that only runs if a condition is correct. It’s a great way to organize events in your code for more complex games!

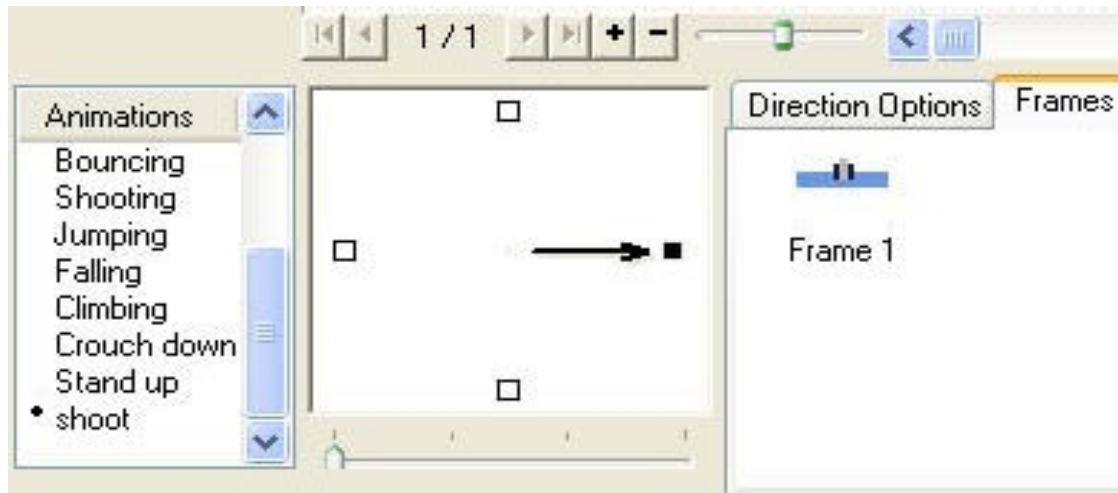
Under the event editor, on a number beside the next condition, select : insert> Group of Events. Call the events Bonus and de-select the “active when frame starts”. We don’t want it to be active when the frame starts, because we want it to only be active when the player gets the bonus.

Our bonus group of events will turn our paddle into a gun and make it shoot bullets out the top. When the gun runs out of bullets, it will return back to being a normal paddle.

The first thing to do is animate our paddle. In the Frame Editor, select the paddle to open the Graphics Editor.

Copy our original paddle. Now on the bottom left under “animations”, below the last item on the list, right click and select “new”, then call it “shoot”. Paste our paddle and then draw a little gun turret onto it.

Now draw an active object for the bullet and call it bullet.



We want to insert 3 different events into our “Bonus” group of events. Event groups are indented in the code, and can be opened or closed to view when looking at the event editor.

The first event is to set the number of bullets to 10. Under the first condition, select the special object, and “Limit Conditions> Only one action while event loops” this means, while the game cycles through this code, it will only do this one code item one time: under the actions, under the paddle, we want to select “Alterable Values” >SET> set value of A to 10.

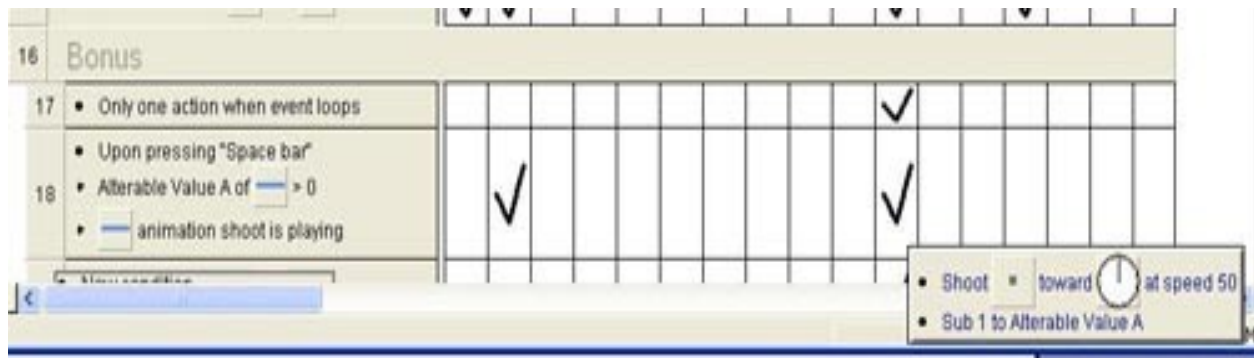
11	Level Three																			
12	• Repeat 10 times																		✓	✓
13	• Collision between and • is facing a direction		✓		✓	✓													✓	✓
14	• Collision between and • is facing a direction		✓								✓								✓	
15	• Collision between and	✓	✓																✓	✓
16	Bonus																			
17	• Only one action when event loops																		✓	
18	• Upon pressing "Space bar" • Alterable Value A of > 0 • animation shoot is playing		✓																✓	

Essentially what we’ve done here is said, we’re going to have a variable that can change, but we’re going to set it to 10 for now.

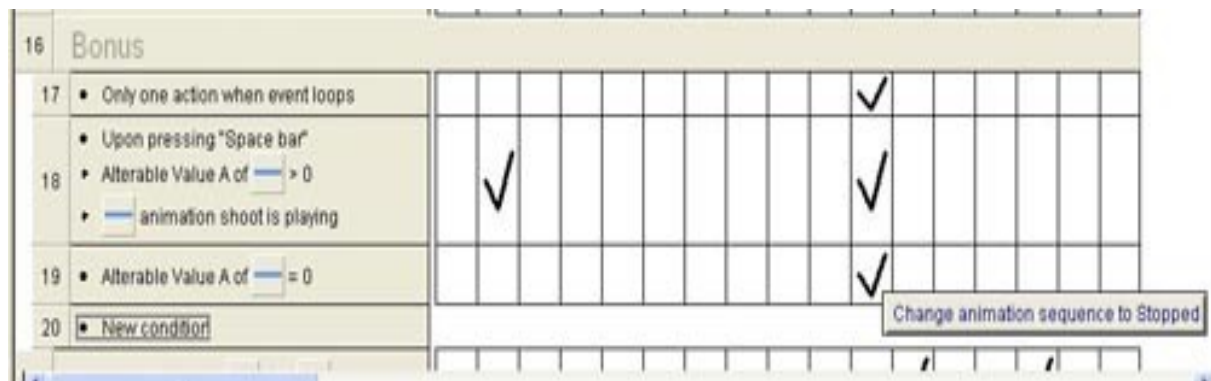
Looking at our pseudocode, the player should be able to shoot by pressing spacebar.

The second condition should be keyboard>upon pressing a key> spacebar. But we need to qualify this condition, because we want to make sure that the correct animation is playing (that is, that the player has changed to “shoot” animation, and that the player has bullets. So insert the additional condition, paddle> Compare alterable value > of A where A is greater than 0. Insert another condition that reads, paddle> animation> which animation of paddle is playing? Shoot.

Now the actions for that condition: play the sound “shoot”, shoot an object (bullet) towards our bricks. We also need to set an action to remove 1 from the number of bullets. So for that, it’s paddle>alterable values> subtract 1 from value A.



Now our final condition: when we run out of bullets, we need to return back to our standard paddle animation.



Condition is paddle> alterable values> compare to alterable values and = 0. (when our variable A has reached 0). Action is : return to the normal animation (Paddle> change animation> stopped).

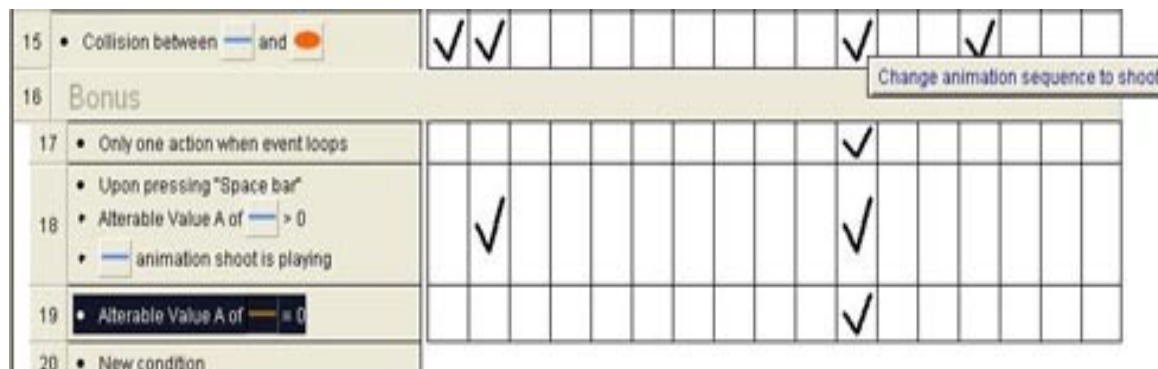
That’s it for our Bonus Event Group.

Now set some final events for our bonus object: Go back under our “level 3” code (above “Bonus”) and set a condition so that when the object gets to the bottom of the screen (therefore there is no collision and we miss the object), destroy the object.

Set another one in the line that reads, “collision between paddle and bonus object” and under special, click on activate group> Bonus.



This means, when we collect the bonus object, this group of events will be activated. So we also need to change the animation when that happens to “shoot”. Under the same line, change the paddle to “Change animation sequence” and change it to “shoot”.



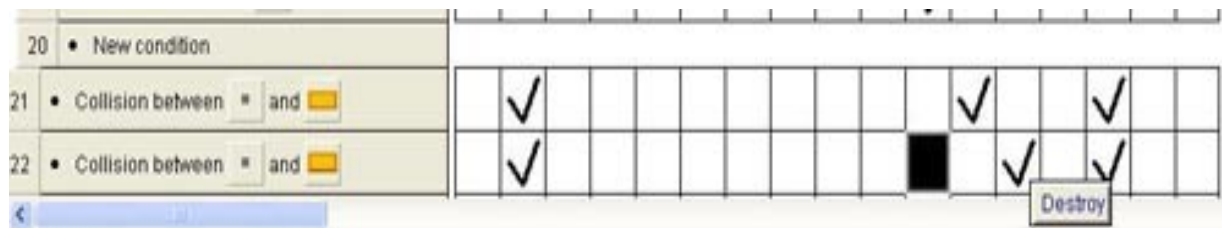
Okay, so our final action is to set what happens when we shoot bullets!

So the bullets are going to collide with the bricks, or they’re going to miss (colliding with the backdrop).

We have two types of bricks, so we need two conditional statements:

- when the bullet collides with “brick”.
- when the bullet collides with “bonusbrick”.

Set the actions for both of these to: destroy brick/destroy bonusbrick (depending on the line). Play the explosion sound. And destroy the bullet, since we want to remove it from gameplay.



When the bullet collides with the backdrop (right click bullet> collision>Backdrop), destroy the bullet.

A final tweak:

We need to correct the line on this frame that reads, “when the last brick is destroyed” and insert another condition so it reads “when the last brick has been destroyed” AND “the last bonusbrick has been

destroyed” then go to next frame. Otherwise, there may be a bonusbrick left and we’re sent to the next screen without having destroyed all the bricks on this screen!

Additional Exercises

1. Add a new level with a new bonus object that increases the size of your paddle.
2. Add yet another level with a bonus object that adds a second ball to the screen.
3. Invent the actions for a new bonus object and create a level for it.