

Pong Tutorial

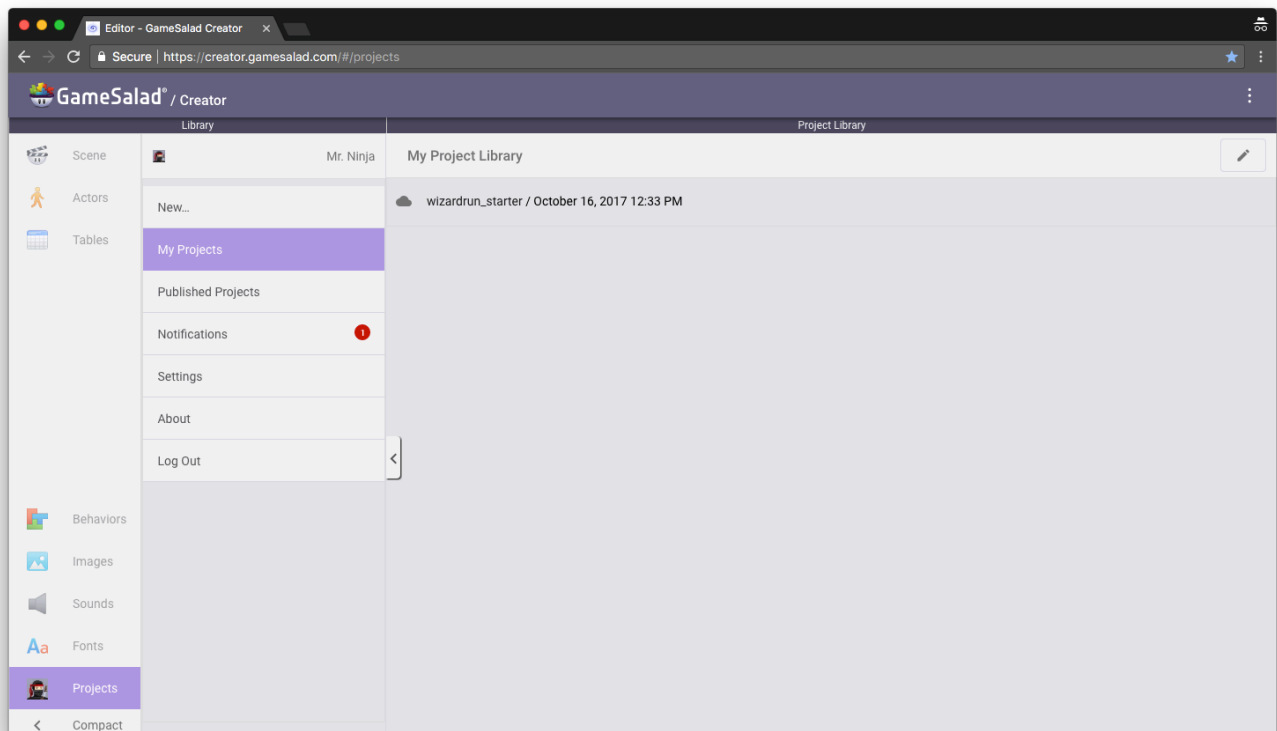
PART 1: SETTING UP YOUR GAME

Adding a Project Name and Setting a Default Scene Size

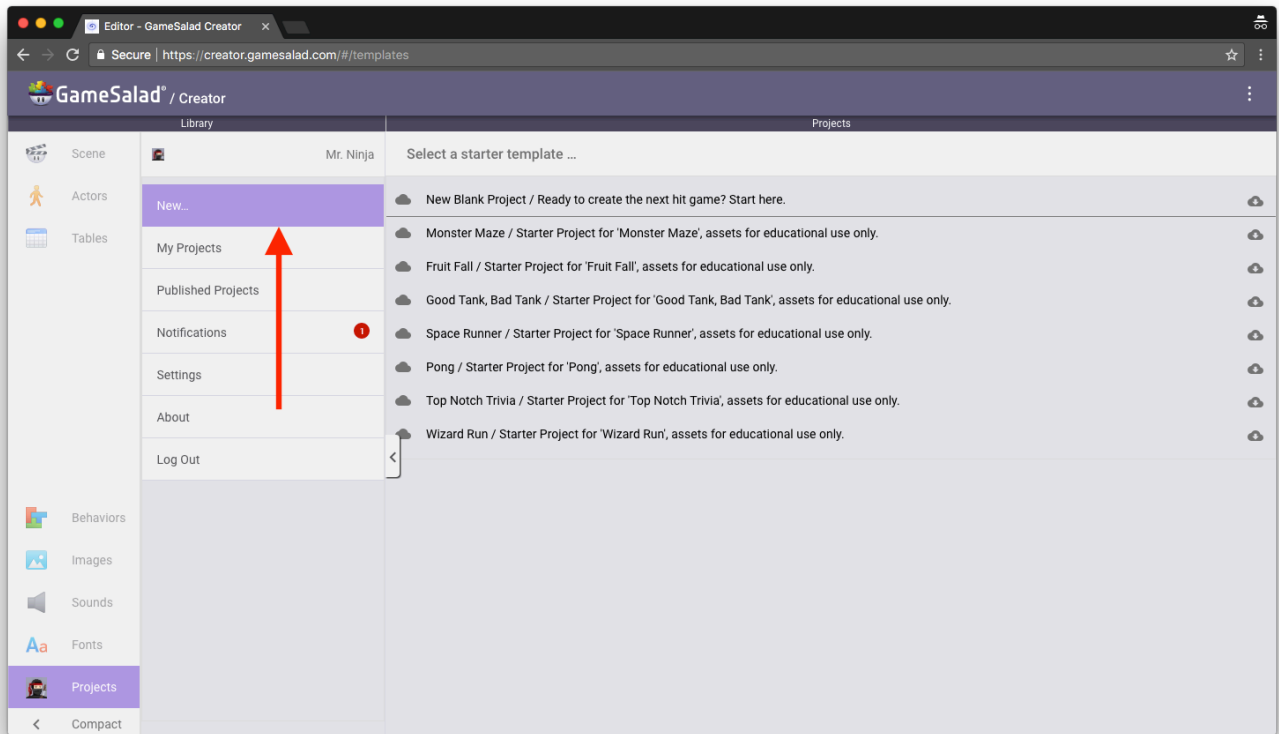
When you first open GameSalad, you'll see the GameSalad Dashboard.

From this window, you can start a new project, open a recent project, or open one of our templates to examine the logic and Behaviors used.

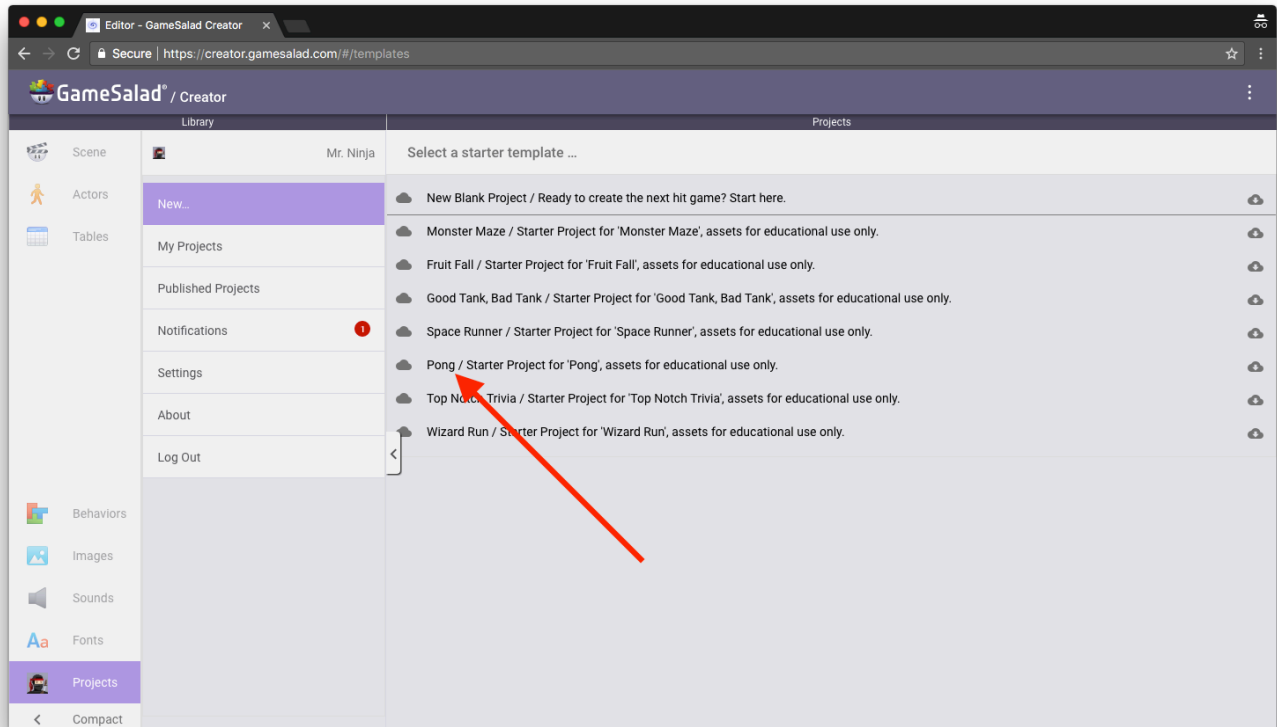
In this tutorial, we're going to start with the Pong starter template. This will have all of our assets (art and sound files) already imported for us!



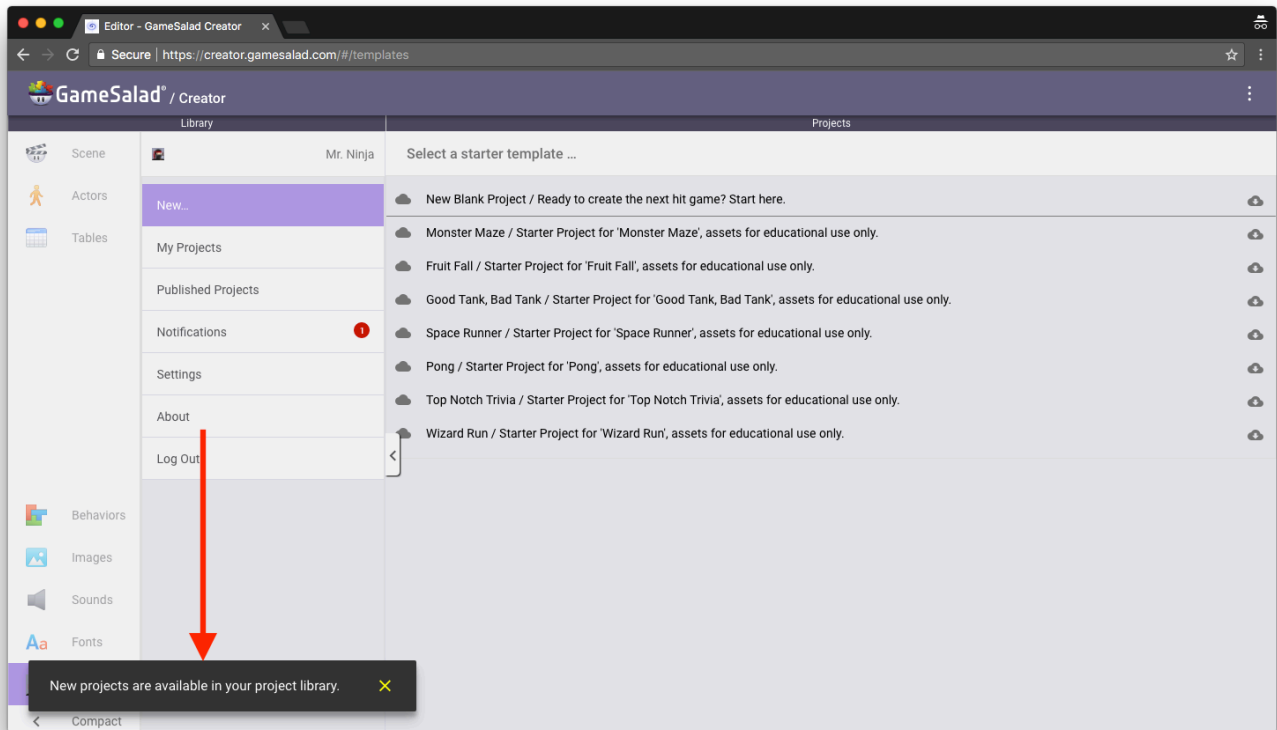
1. Click on 'New...' in the upper left hand side of the window to see the list of templates you can start from.



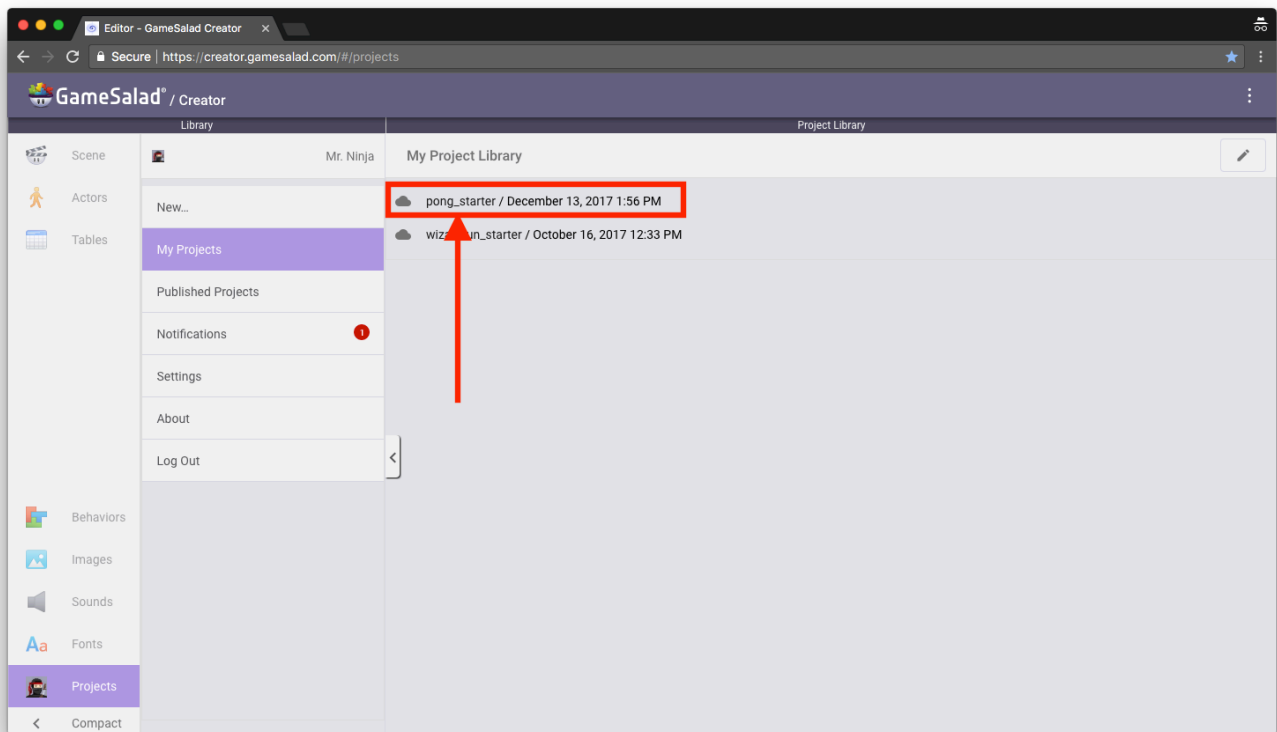
2. In the list of templates shown, click on the 'Pong' starter project. GameSalad will import this into your Project Library! You should see a message pop up in the bottom left of the browser to inform you that your template is being prepared.



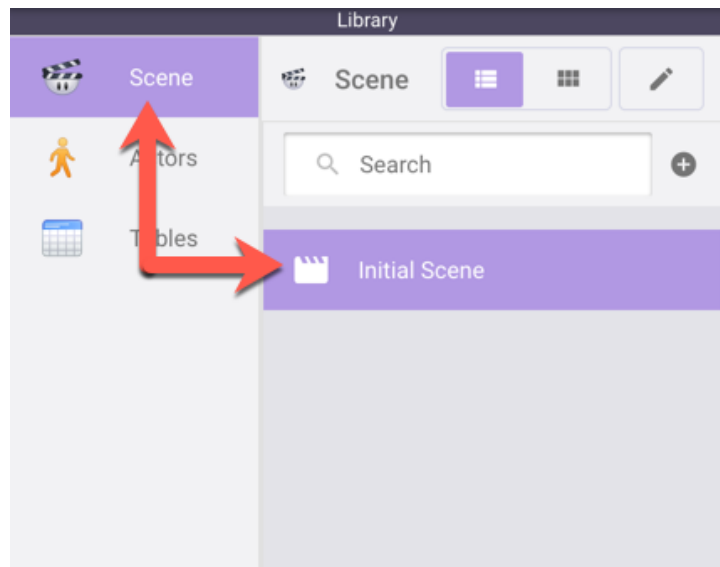
3. When it's done, you should see another message telling you that you have new projects available in your Project Library.



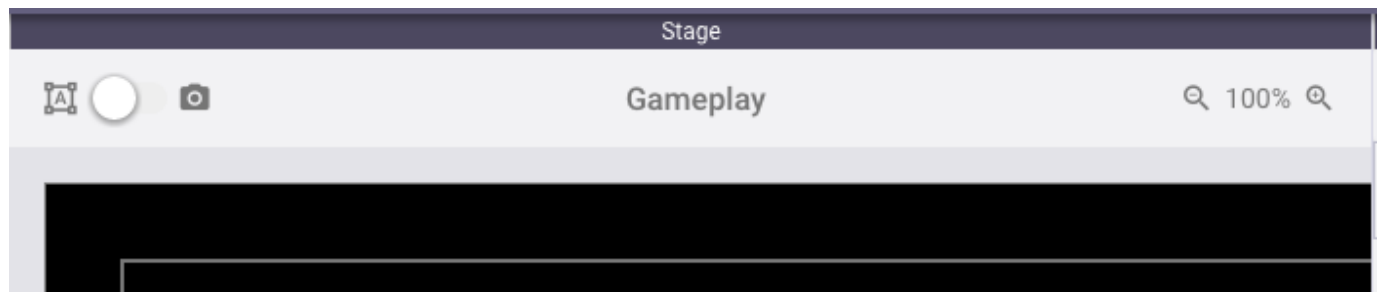
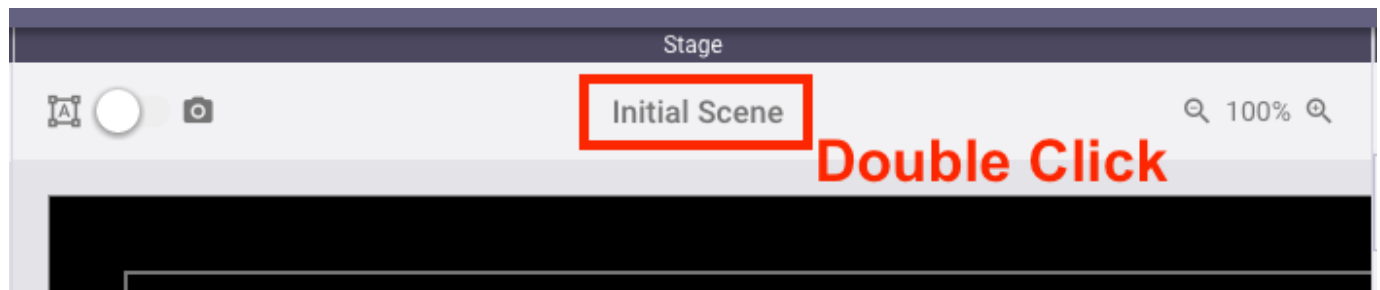
4. To access the Project Library, click on 'My Projects' from the list of tabs on the left. You should now see a project listed with the name 'pong' at the top of the list.



5. Click on the project to open it in GameSalad! When you open your project, you will see the Scene Editor.
6. Click on the Scenes tab in the Library. Notice that the 'Initial Scene' is already active in the Library.



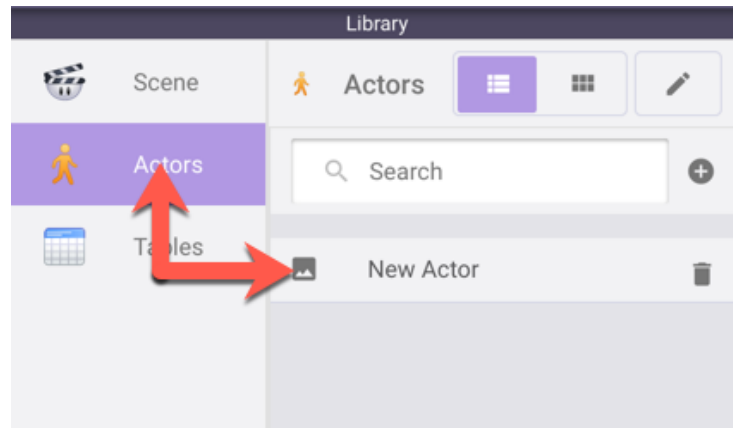
7. Rename the scene by double clicking on the text "Initial Scene" in the Stage Bar. We're going to call this scene "Gameplay".



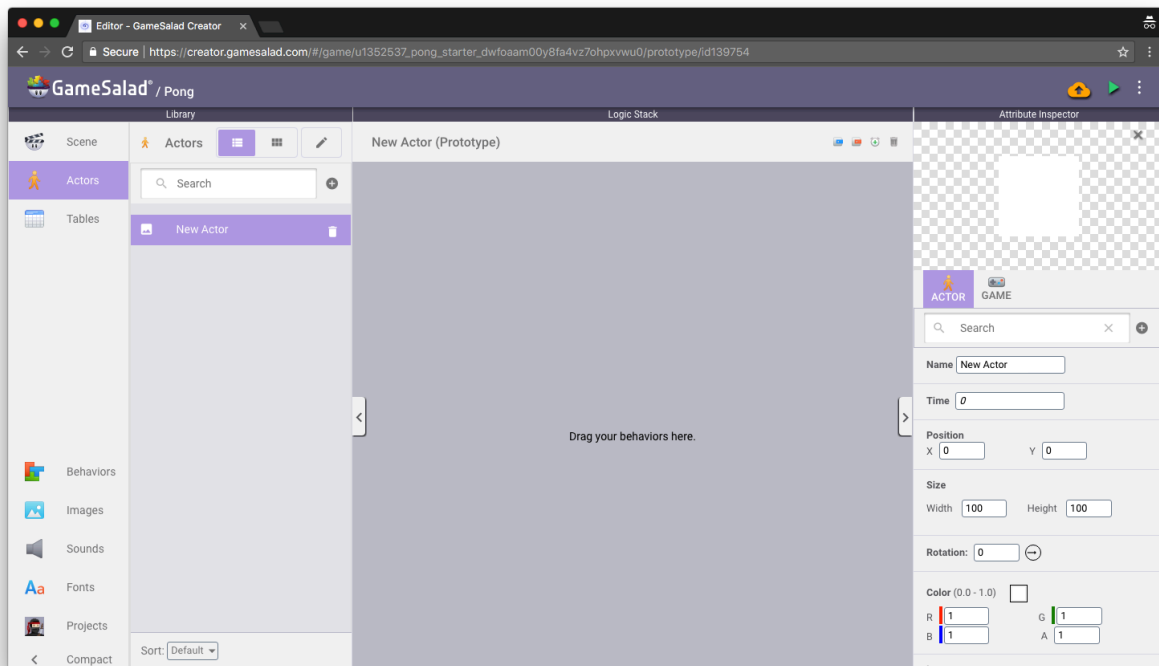
Adding a Background

Next, let's add a background to our project.

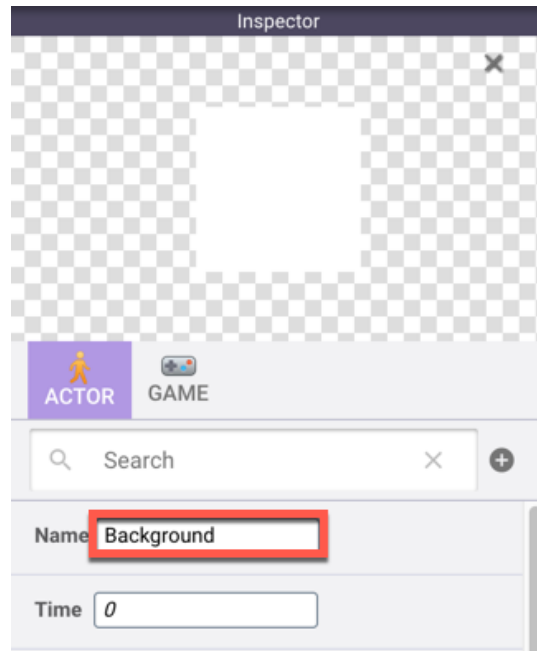
1. Click the Actors tab in the Library.
2. Click the '+' icon to the right of the Search Bar to create a new actor.



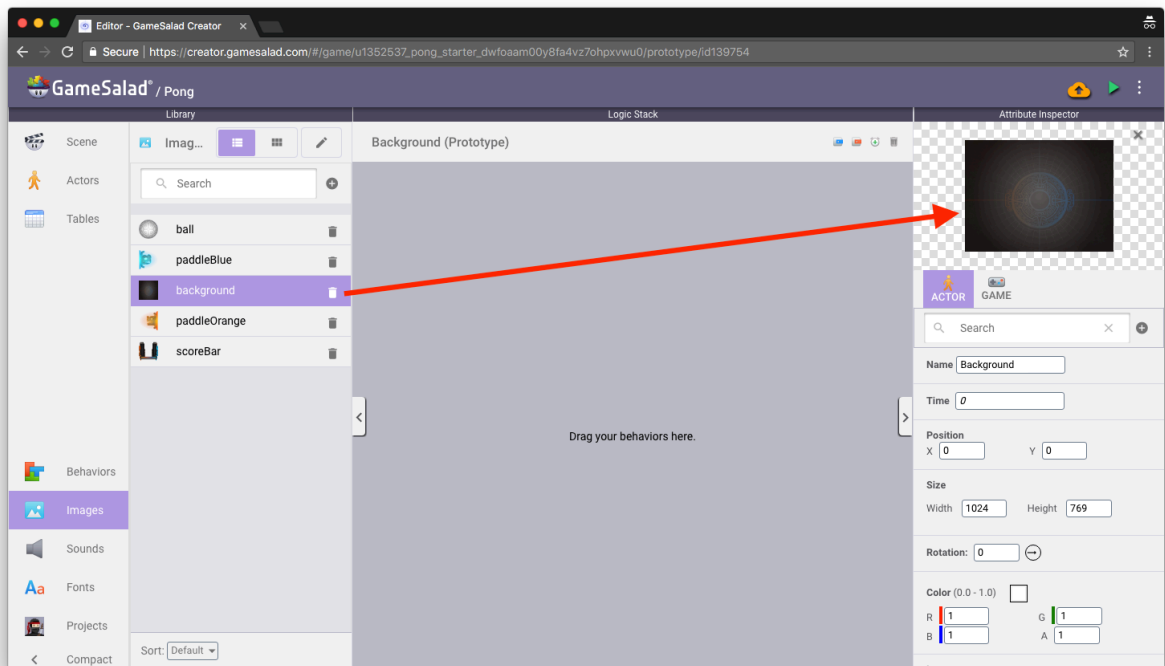
3. Click on the newly created actor to navigate to the Actor Editor for this specific actor.



4. In the Inspector on the right hand side, locate the 'Name' Attribute and change the name of the actor to 'Background'.

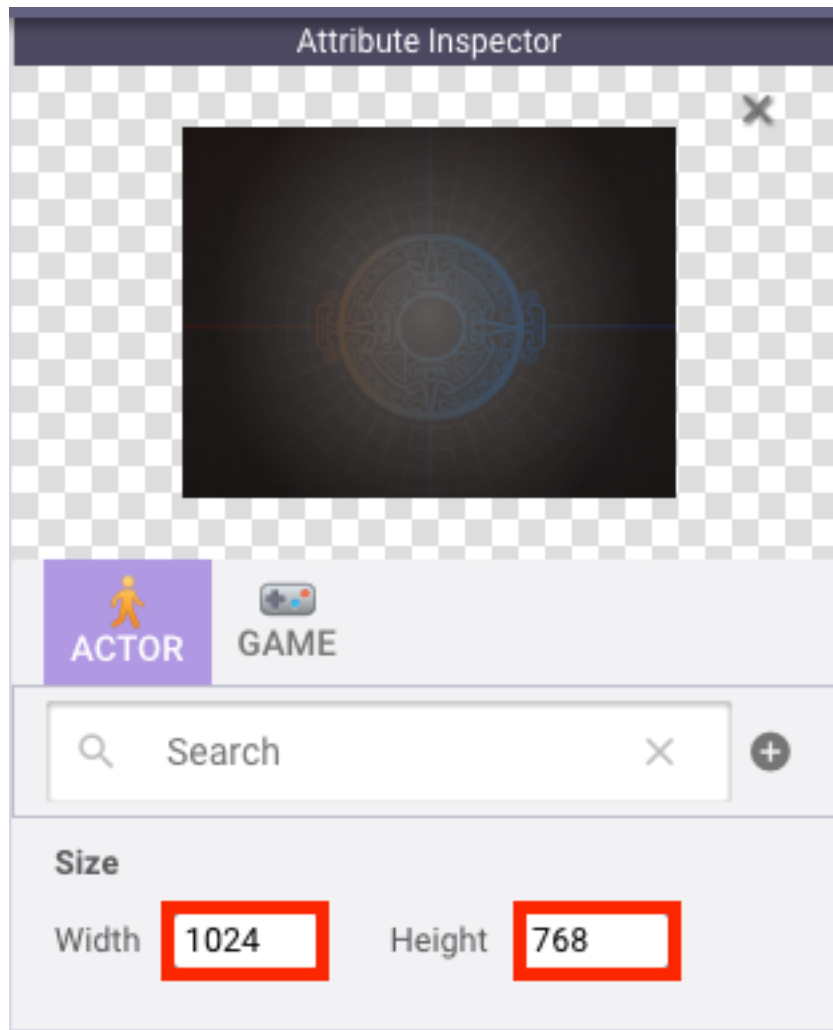


5. Click the Images tab in the Library and locate the “background” image.
6. Drag the image on top of the white square in the top right of the Inspector. Release the mouse button to apply the image to the actor.

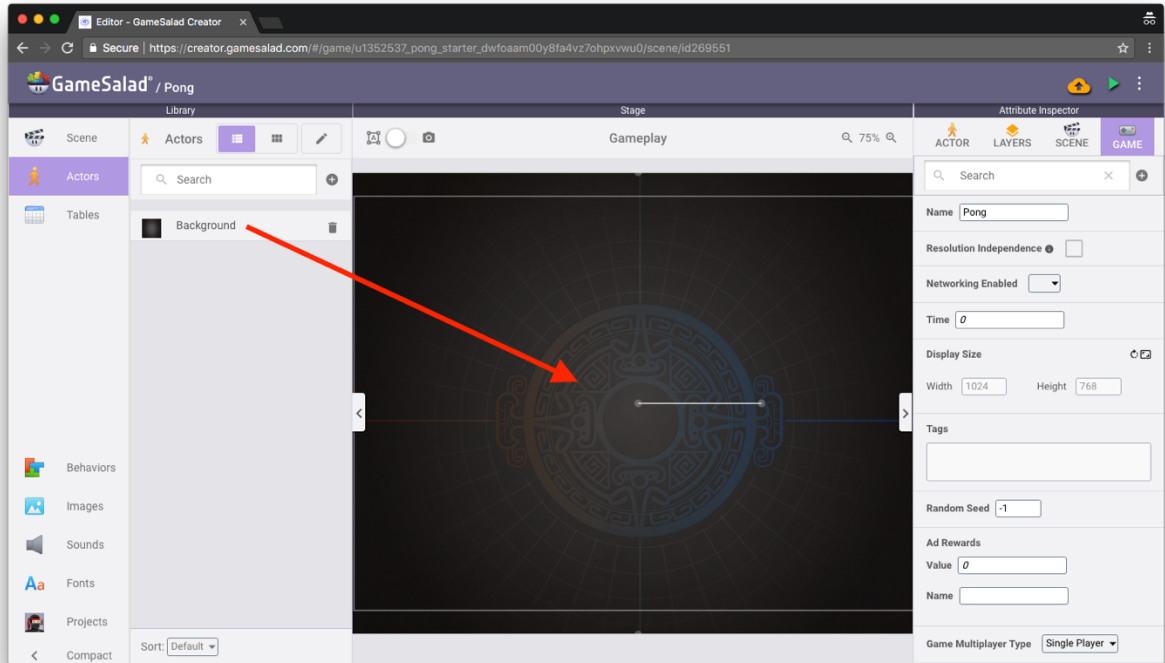


7. Locate the ‘Size’ Attributes in the Inspector.

8. Make sure the 'Width' is '1024' and the 'Height' is '768'.



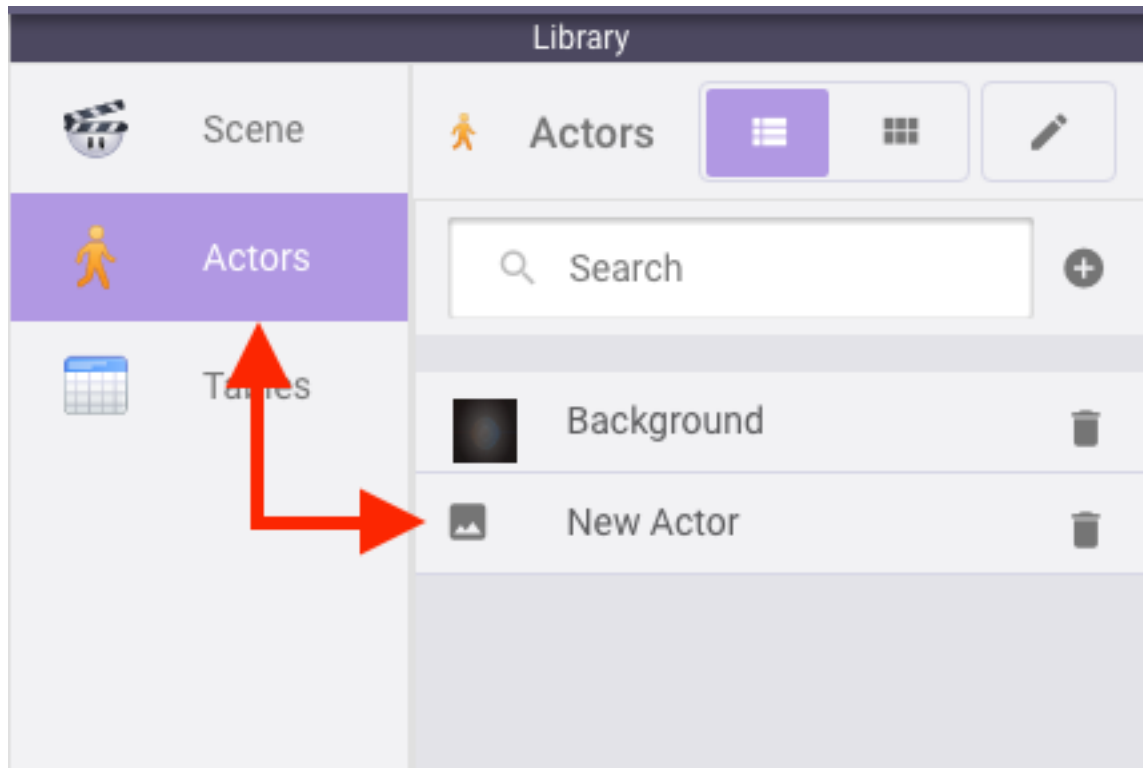
9. Navigate to the Scene Editor for 'Gameplay' by clicking the Scenes tab in the Library and then clicking on the 'Gameplay' Scene.
10. Click the Actors tab in the Library to activate it. Click and drag the 'Background' actor onto the Scene and position it so that it is centered on the scene. NOTE: If you have a small computer scene, you can use the Zoom In/Out button in the top right of the Stage Bar to zoom out of the Scene to see it in its entirety.



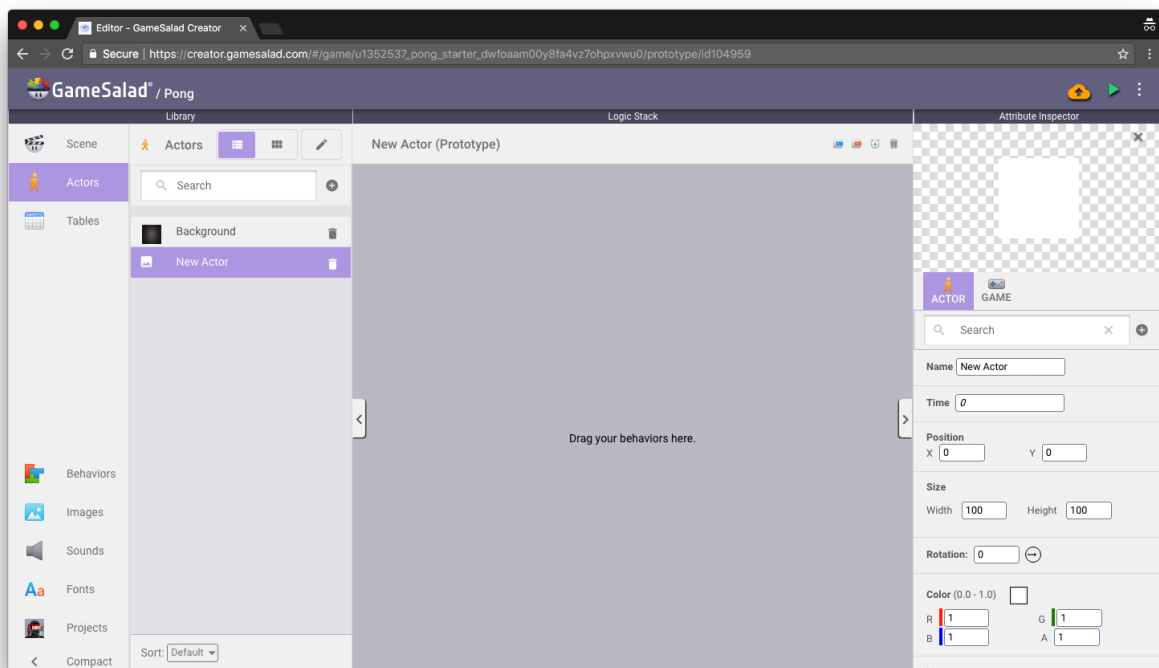
Creating the Paddles

Next, let's create the paddles (what the players of our game will be controlling).

1. Click the Actors tab in the Library.
2. Click the '+' icon to the right of the Search Bar to create a new actor.

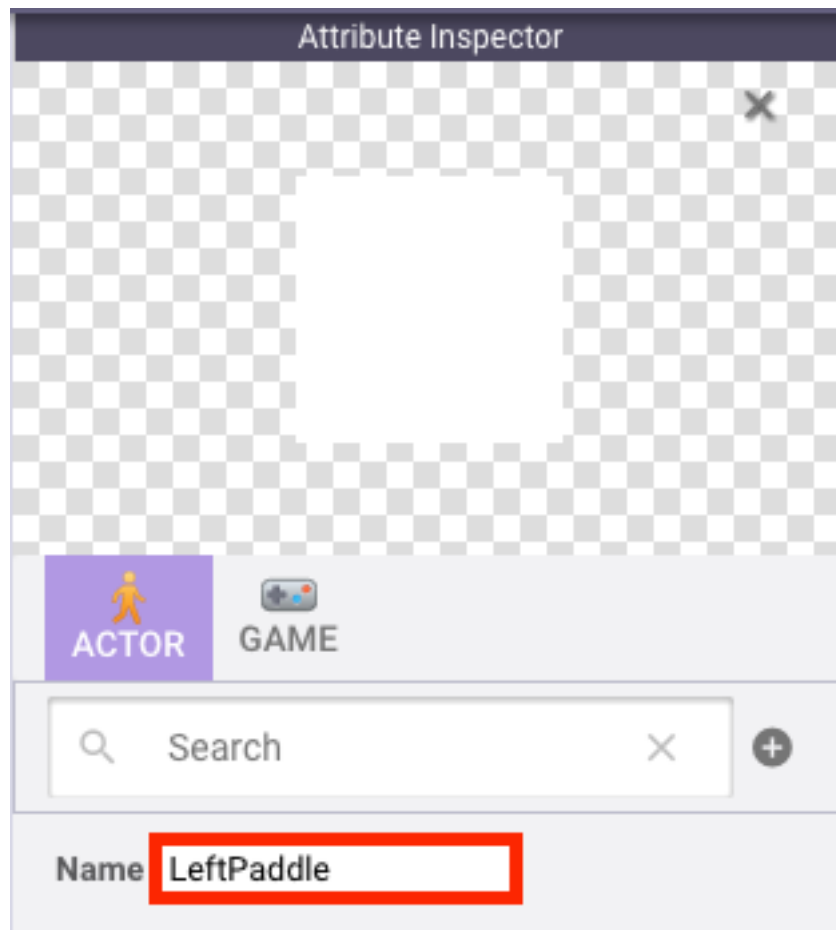


3. Click on the newly created actor to navigate to the Actor Editor for this specific actor.

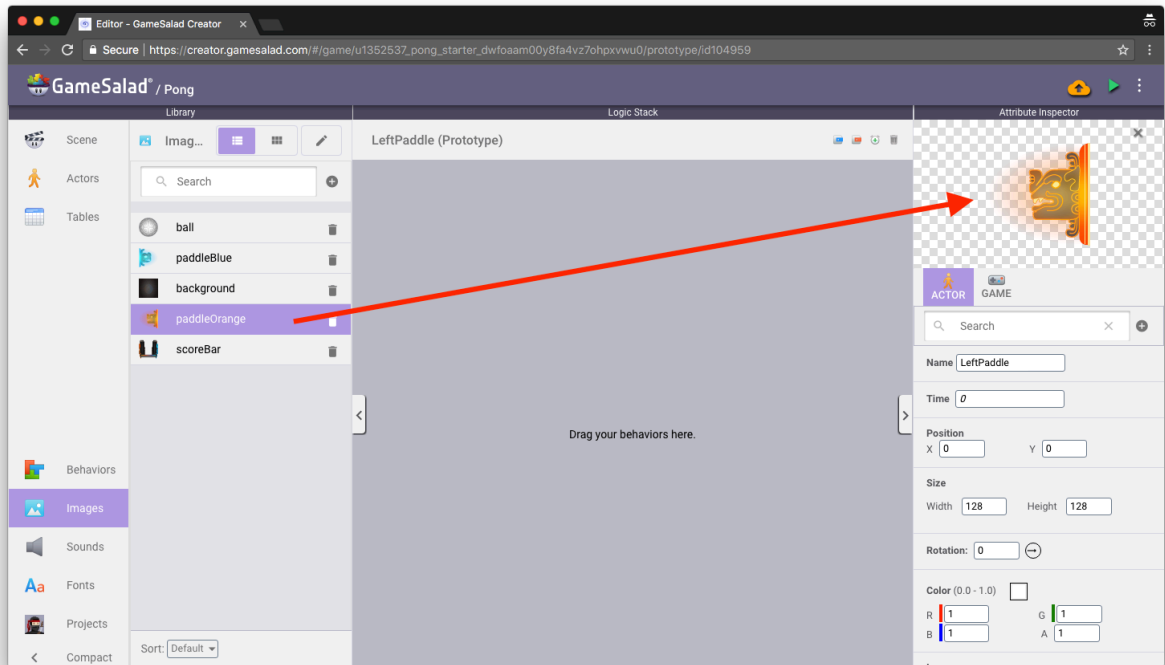


4. In the Inspector on the right hand side, locate the 'Name' Attribute and change the name of the actor to

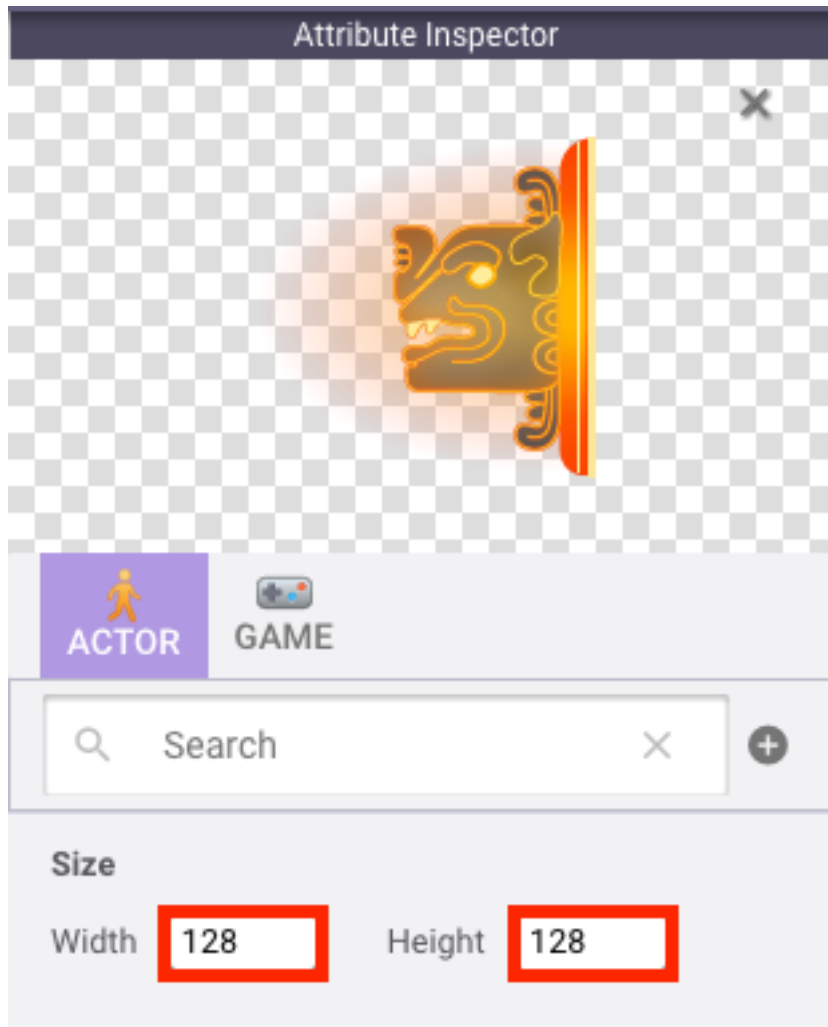
‘LeftPaddle’.



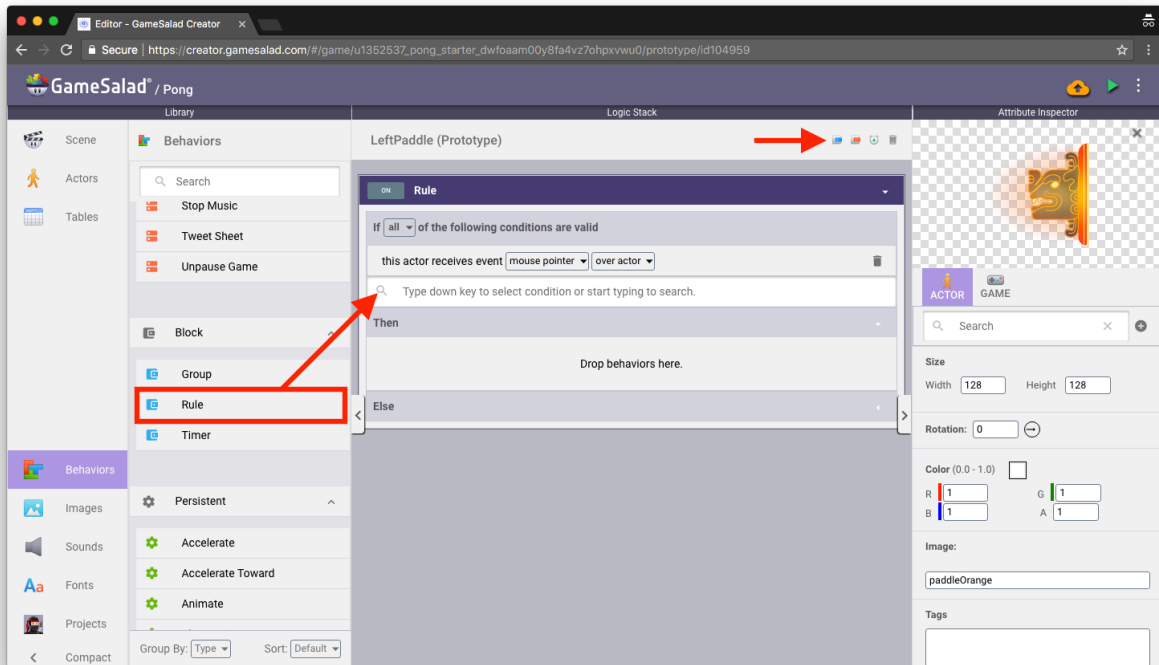
5. Click the Images tab in the Library and locate the “paddleOrange” image.
6. Drag the image on top of the white square in the top right of the Inspector. Release the mouse button to apply the image to the actor.



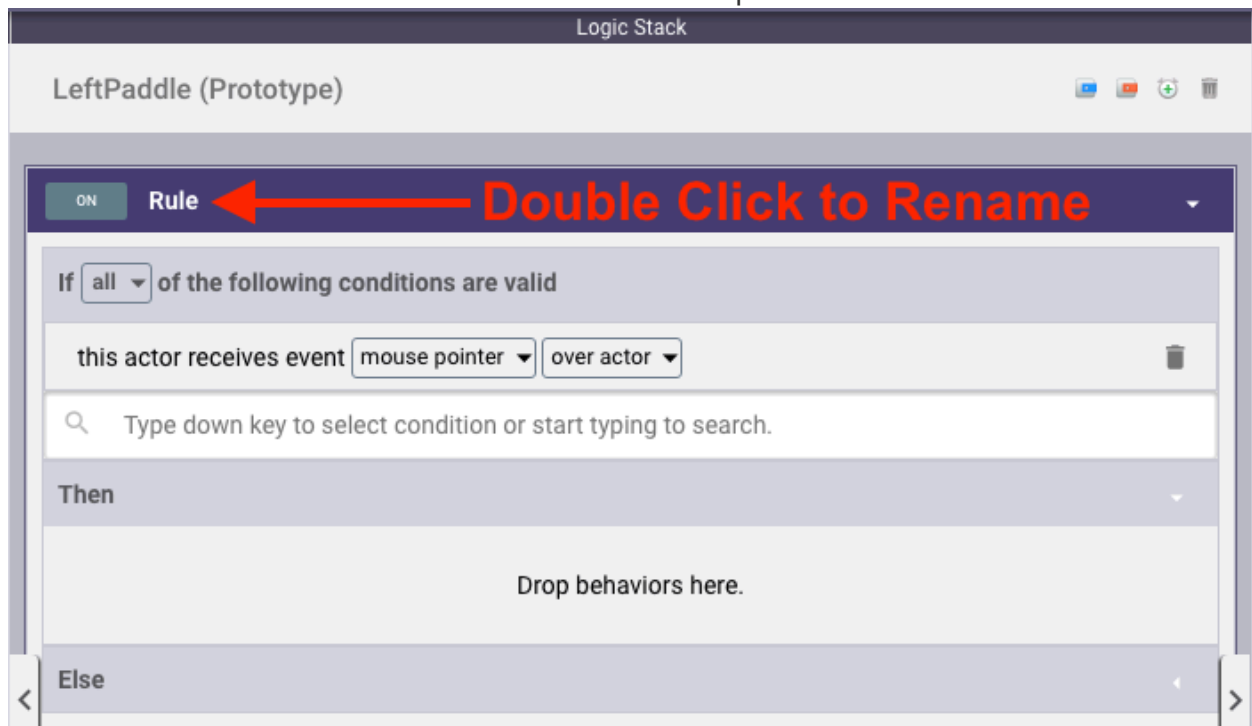
7. Locate the 'Size' Attributes in the Inspector.
8. Make sure the 'Width' is '128' and the 'Height' is '128'.

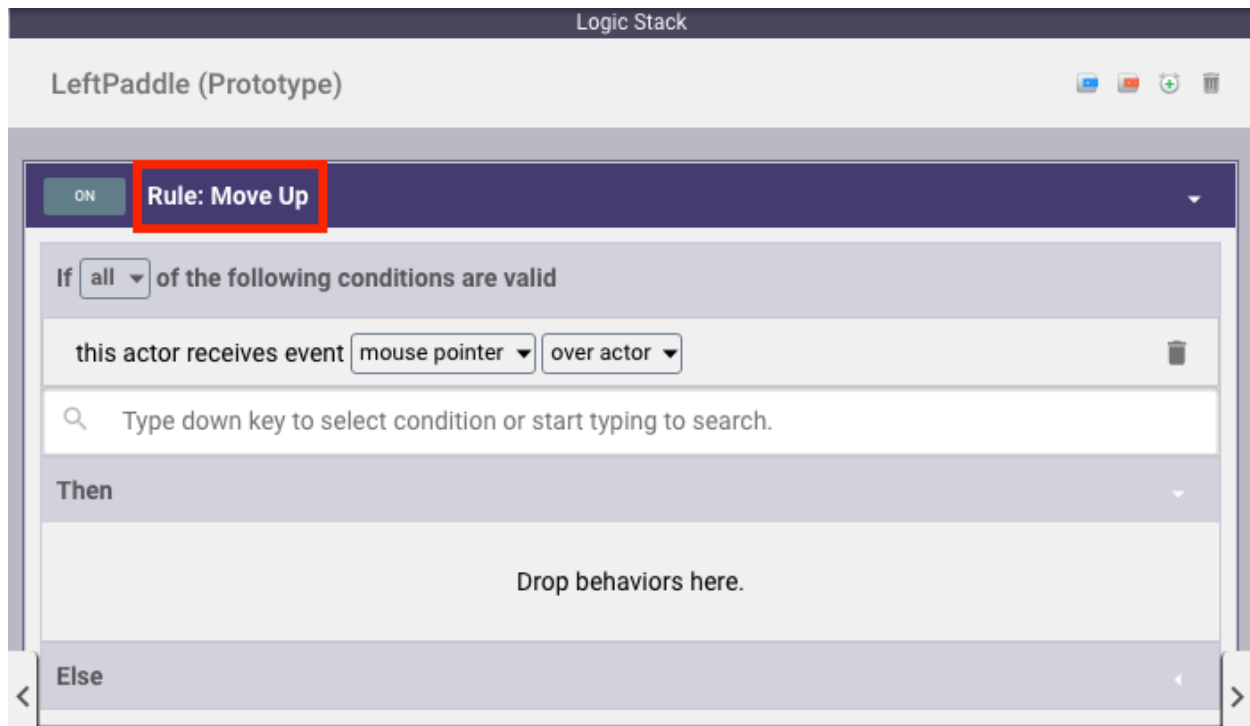


9. Add a rule to the leftPaddle actor. This can be done in two ways:
 - a. Locate the 'Rule' behavior inside the Behaviors tab of the Library, and drag it into the Logic Stack.
 - b. Click the blue 'Add Rule' button located in the Logic Stack Bar.



10. Rename the newly created Rule by double clicking on the word 'Rule' in the Rule header. Rename it to 'Rule: Move Up'.





- Click on the first dropdown button in the rule (currently with 'mouse pointer' selected) and choose the 'keyboard key' option.



- Click in the blank field and press the up arrow key to set it as the keyboard key to be used in the condition.



- Click on the Behaviors tab in the Library to open the prebuilt Behaviors available to us. Locate the 'Move' Behavior in the list.

Scene

Actors

Tables

Behaviors

Images

Sounds

Fonts

Projects

Compact

Library

Behaviors

Search

Collide

Constrain Attribute

Control Camera

Display Text

Emit Particles

Interpolate

Move

Move To

Note

Replicate

Rotate

Rotate To Angle

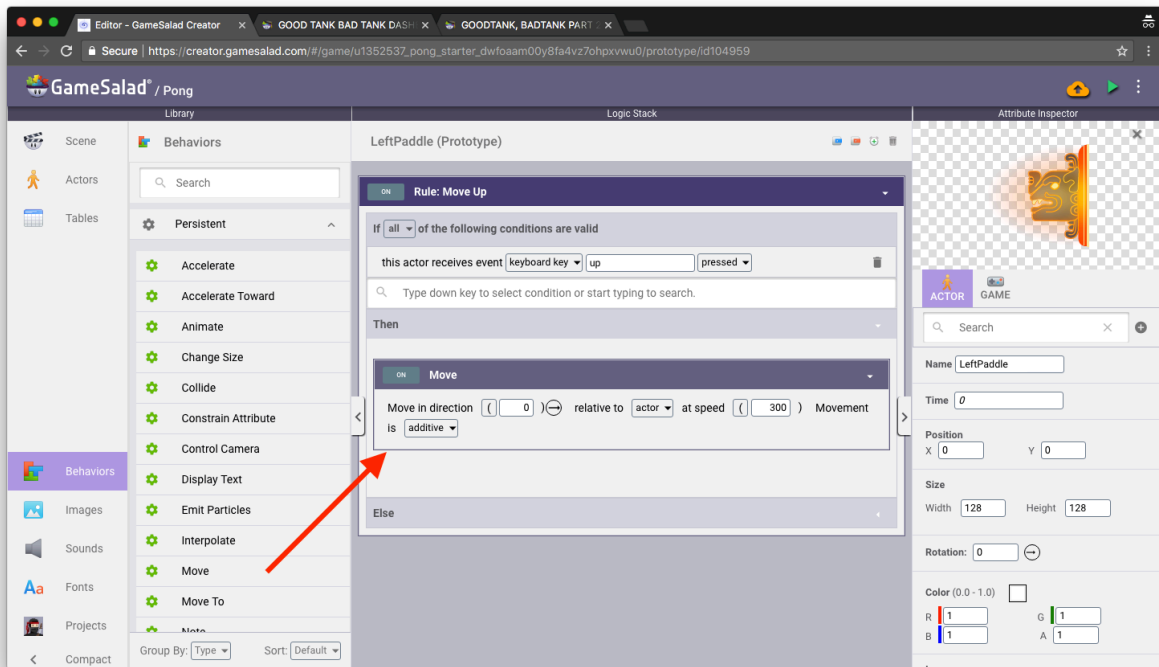
Rotate To Position

Show Banner Ad

Group By: Type

Sort: Default

14. Drag the 'Move' Behavior into the 'Then' section of your Rule.

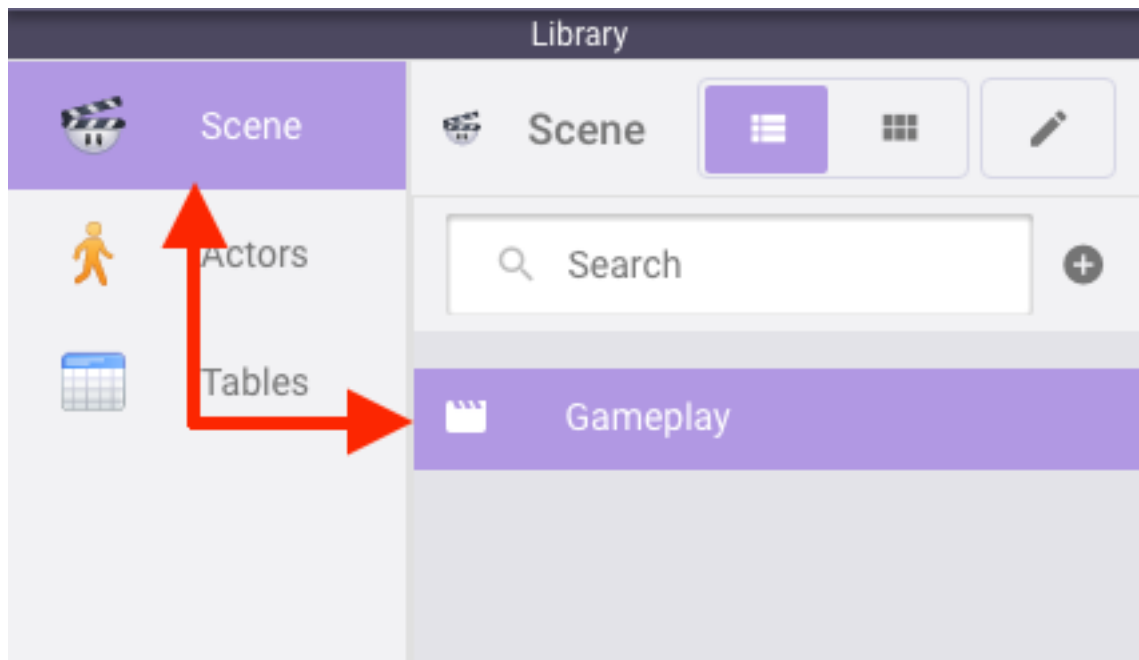


15. Set the direction for the Move Behavior to '90'. This will cause the LeftPaddle Actor to move upward when the up arrow key is pressed.

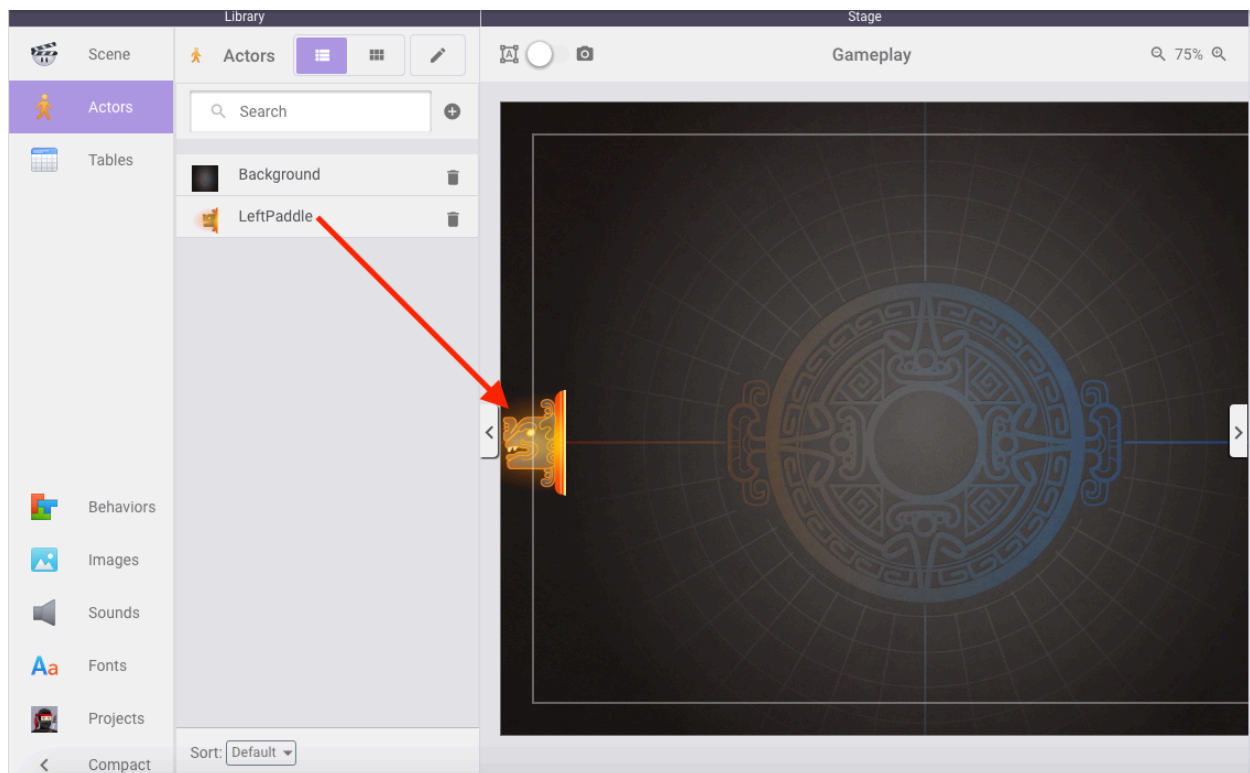


16. Now that we have some logic in our 'LeftPaddle' actor, let's add it to the scene.

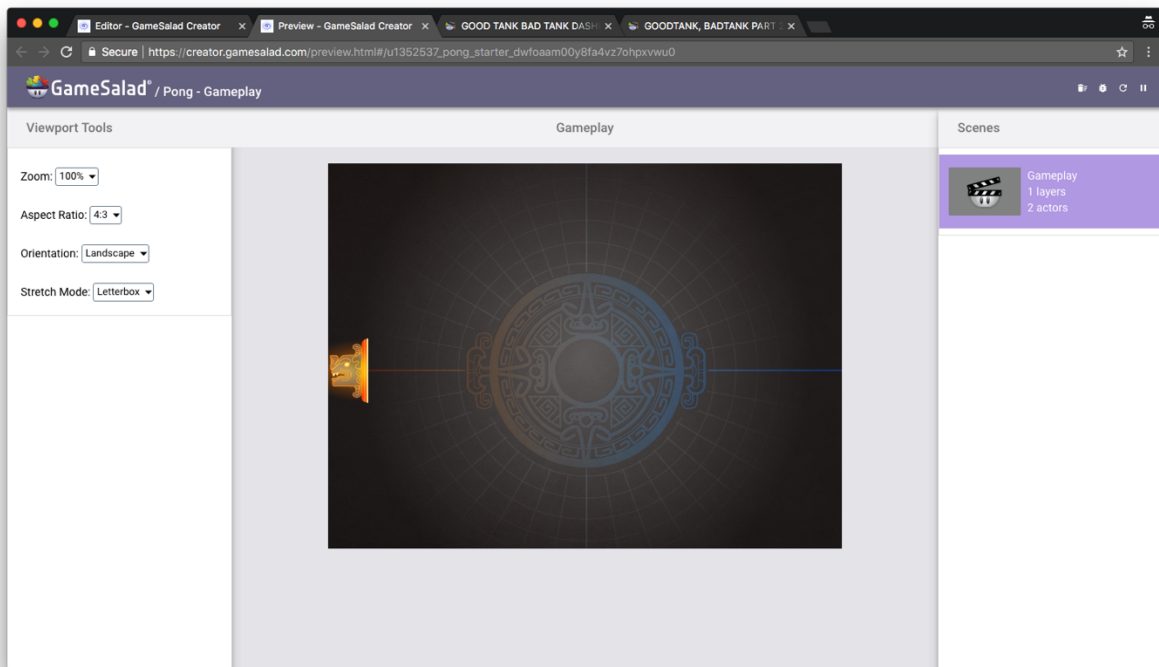
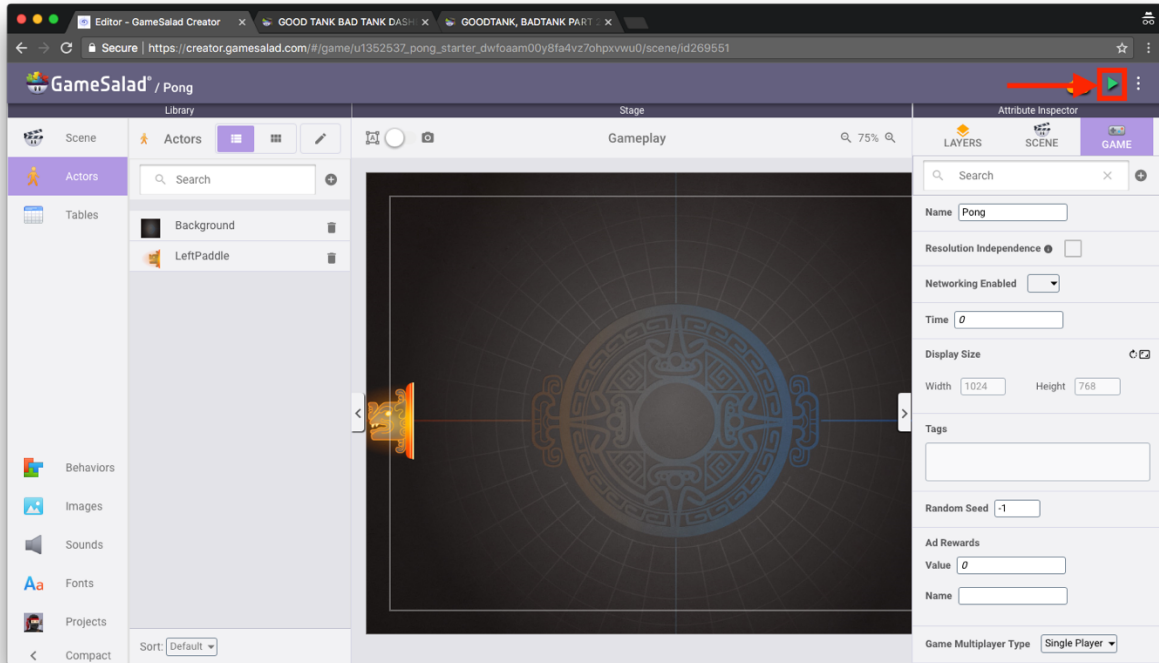
17. Click on the Scene tab in the Library. Click on the 'Gameplay' scene to open the Scene Editor.



18. Click on the Actors tab in the Library if it is not already active. Drag the 'LeftPaddle' actor onto the scene and position it towards the left side.



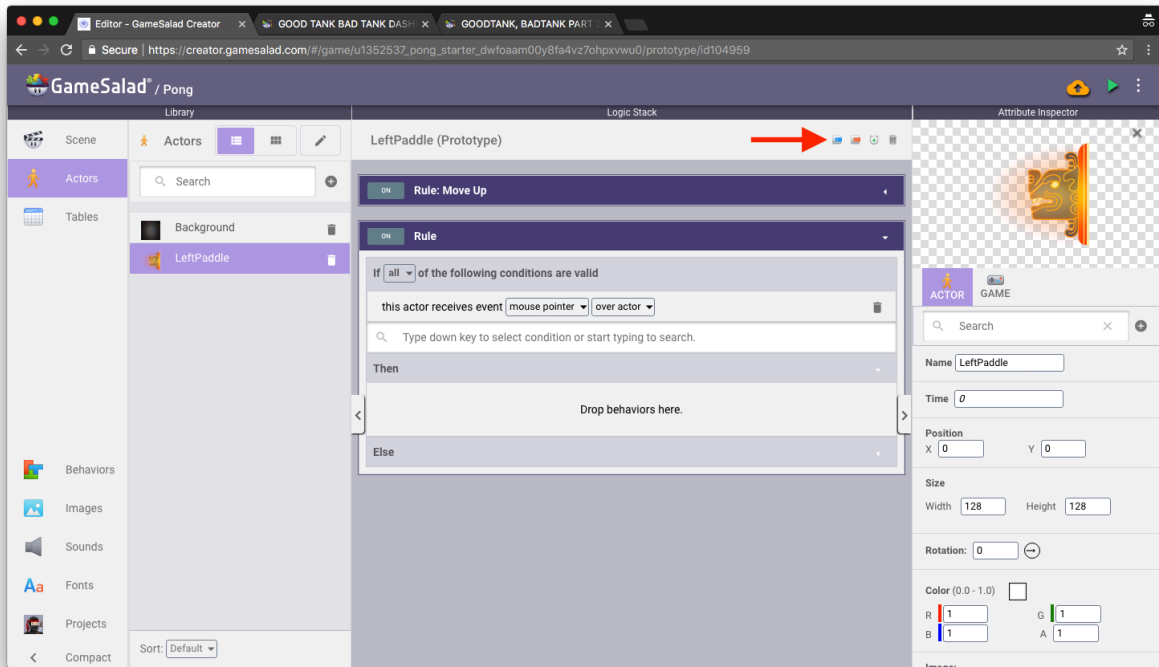
19. Preview the game by clicking the green Preview Game button in the top right of the tool. This will open the 'Preview Player'.



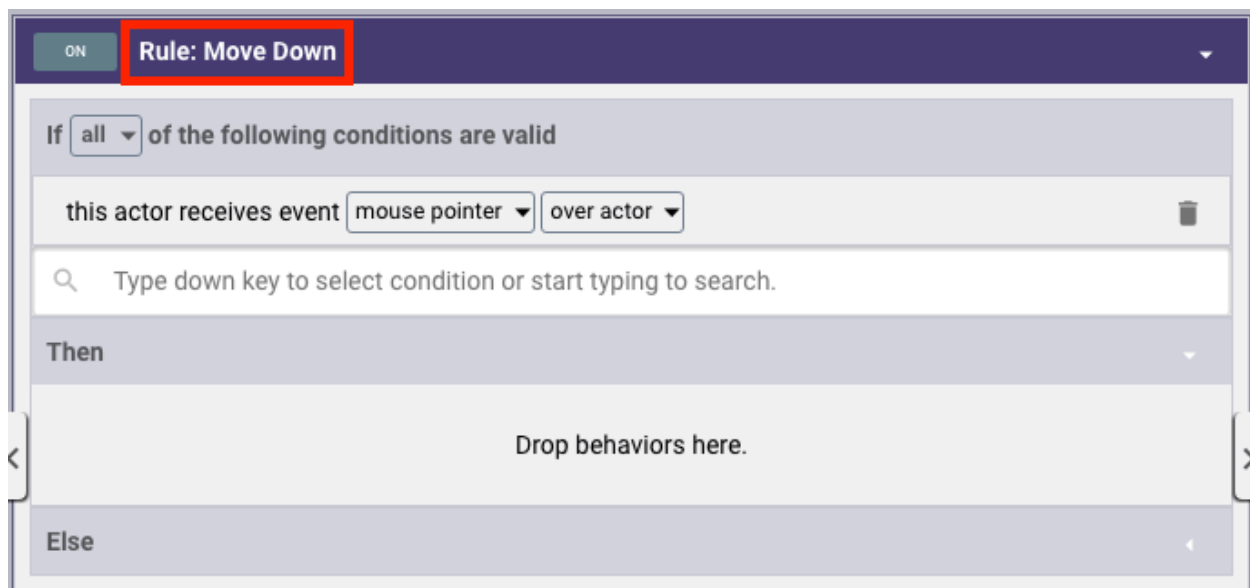
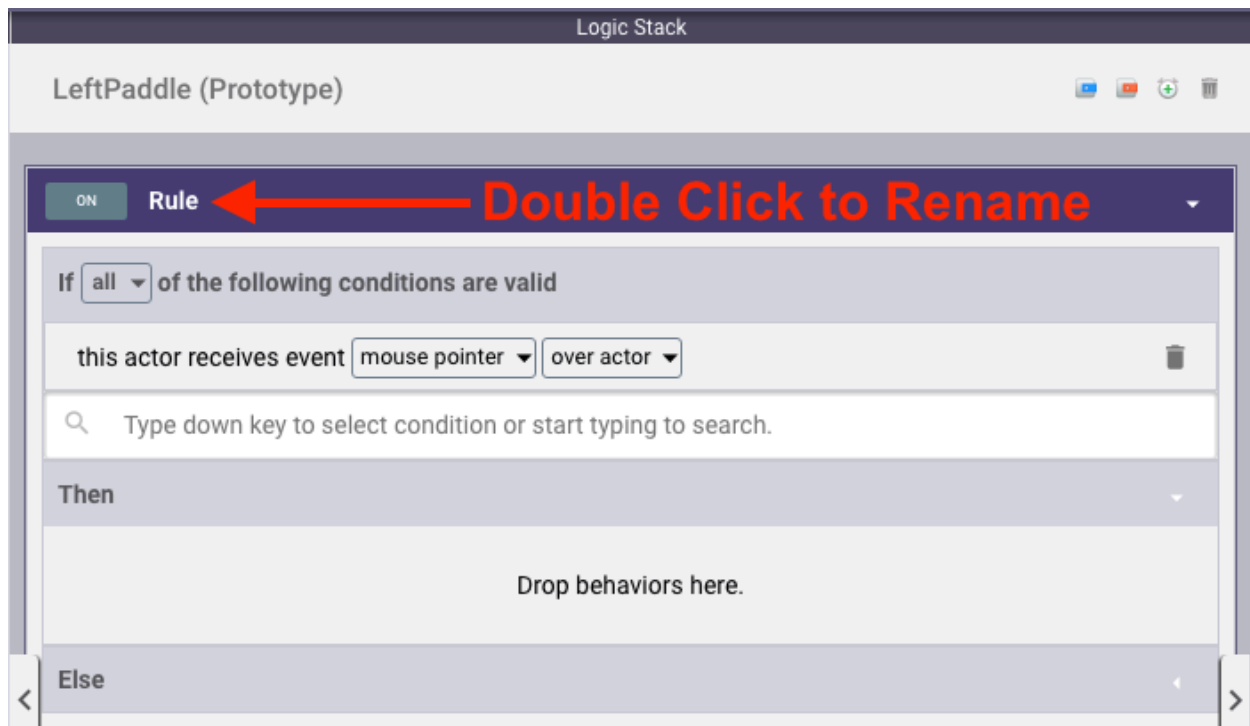
20. Press the up arrow key to make sure that the paddle moves up correctly.

Next let's add downward movement for the paddle. This will be very similar to the Move Up rule we just created.

21. Click on the Actors tab in the Library and then click on the 'LeftPaddle' actor to open the Actor Editor.
22. Add a rule to the LeftPaddle actor by clicking the blue 'Add Rule' button in the upper right of the Logic Stack Bar.



23. Rename the newly created Rule by double clicking on the word 'Rule' in the Rule header. Rename it to 'Rule: Move Down'.



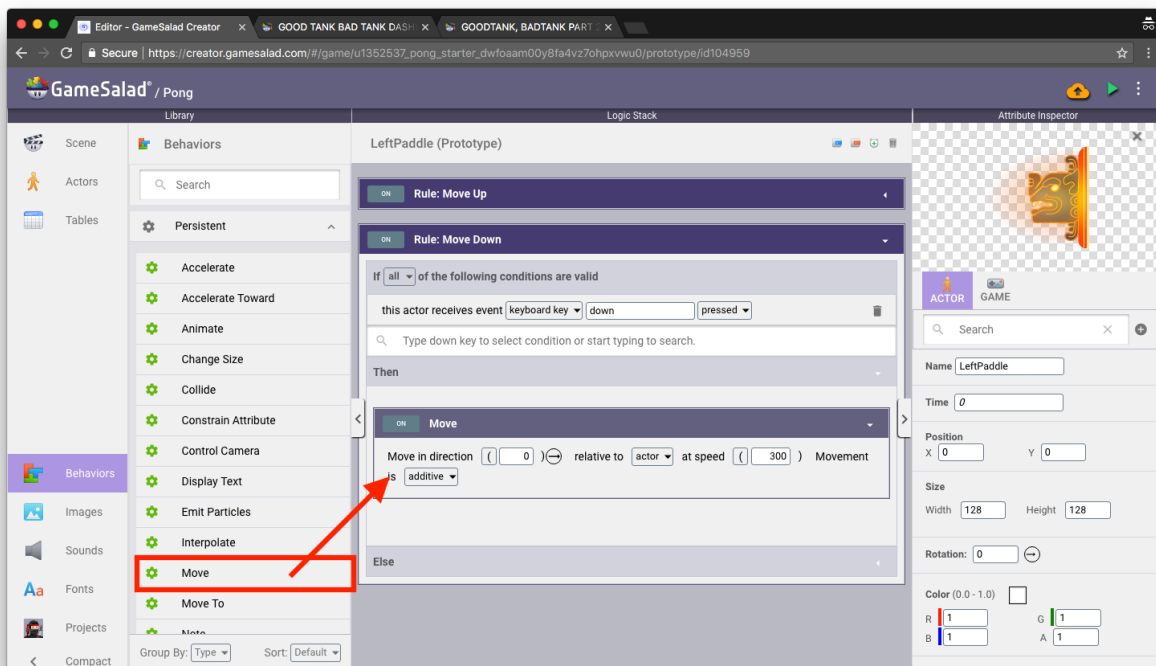
24. Click on the first dropdown button in the rule (currently with 'mouse pointer' selected) and choose the 'keyboard key' option.



25. Click in the blank field and press the down arrow key to set it as the keyboard key to be used in the condition.



26. Click on the Behaviors tab in the Library to open the prebuilt Behaviors available to us. Locate the 'Move' Behavior in the list.
27. Drag the 'Move' Behavior into the 'Then' section of your Move Down Rule.



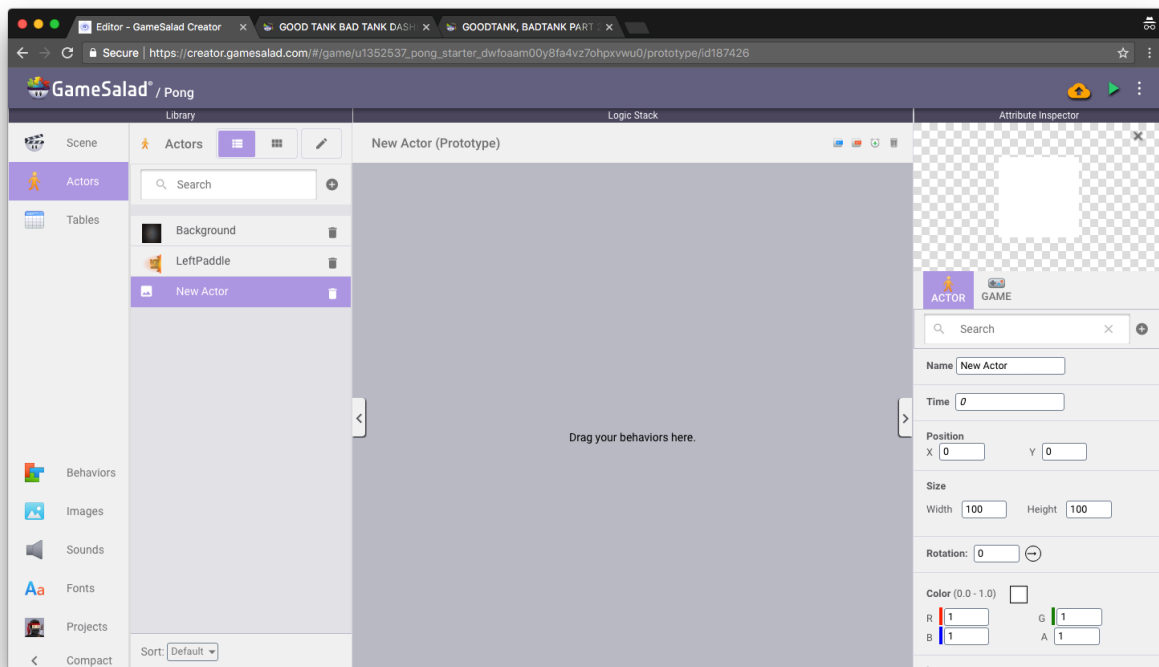
28. Set the direction for the Move Behavior to '270'. This will cause the LeftPaddle Actor to move upward when the down arrow key is pressed.



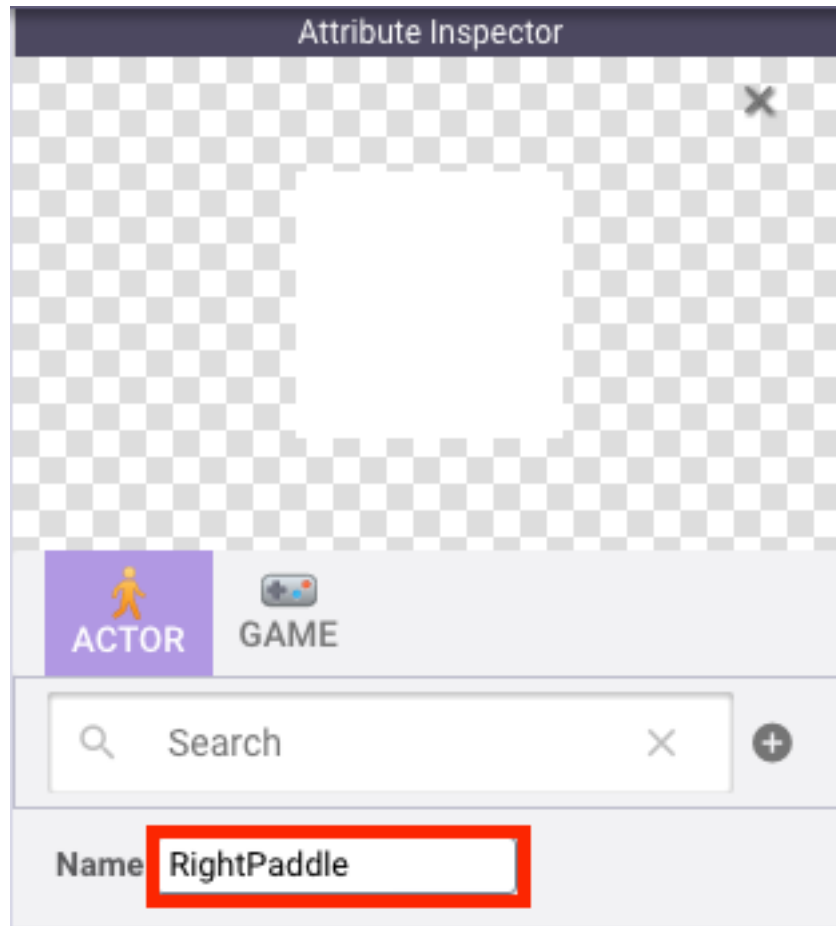
29. Preview the game again by clicking the green preview button in the upper right of the Actor Editor to make sure that the paddle moves up and down correctly when the up and down arrow keys are pressed.

Next let's create the RightPaddle actor.

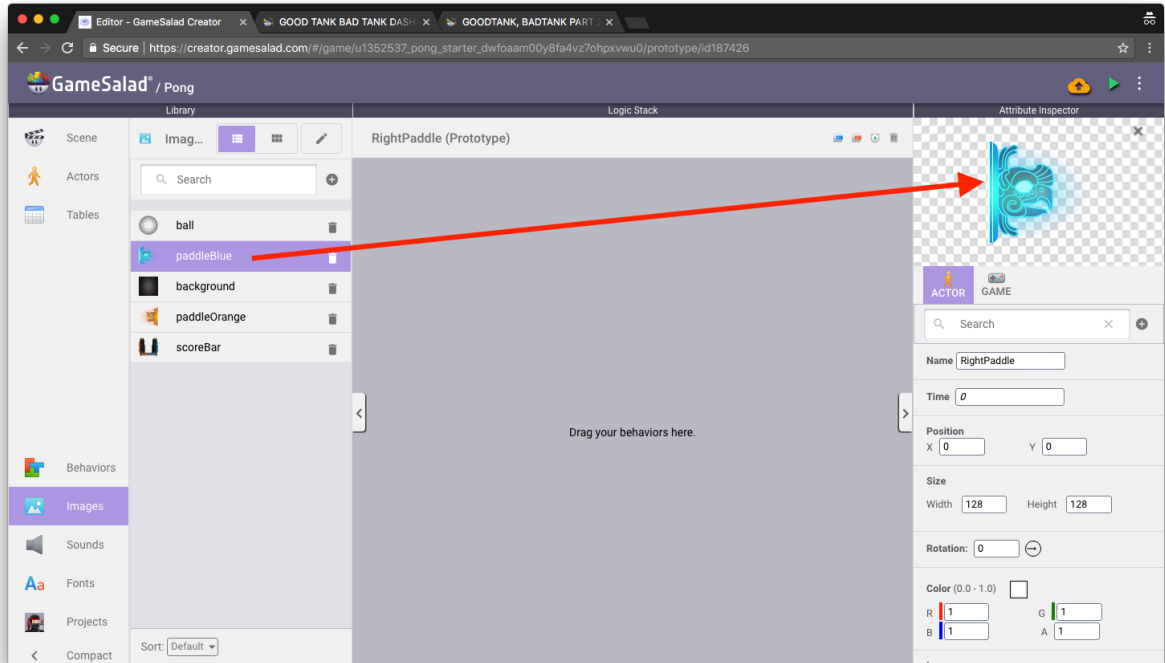
30. Click the Actors tab in the Library and click the '+' icon to the right of the Search Bar to create a new actor.
31. Click on the newly created actor to navigate to the Actor Editor for this specific actor.



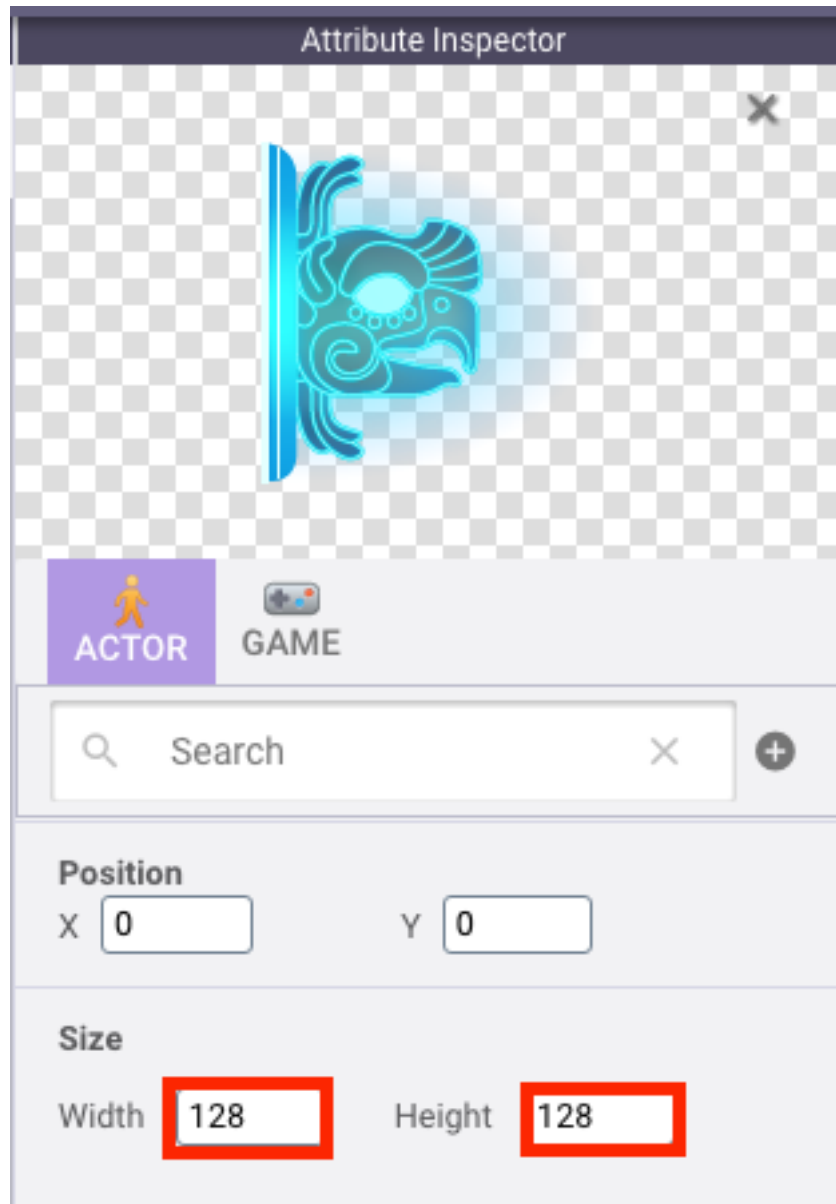
32. In the Inspector on the right hand side, locate the 'Name' Attribute and change the name of the actor to 'PaddleRight'.



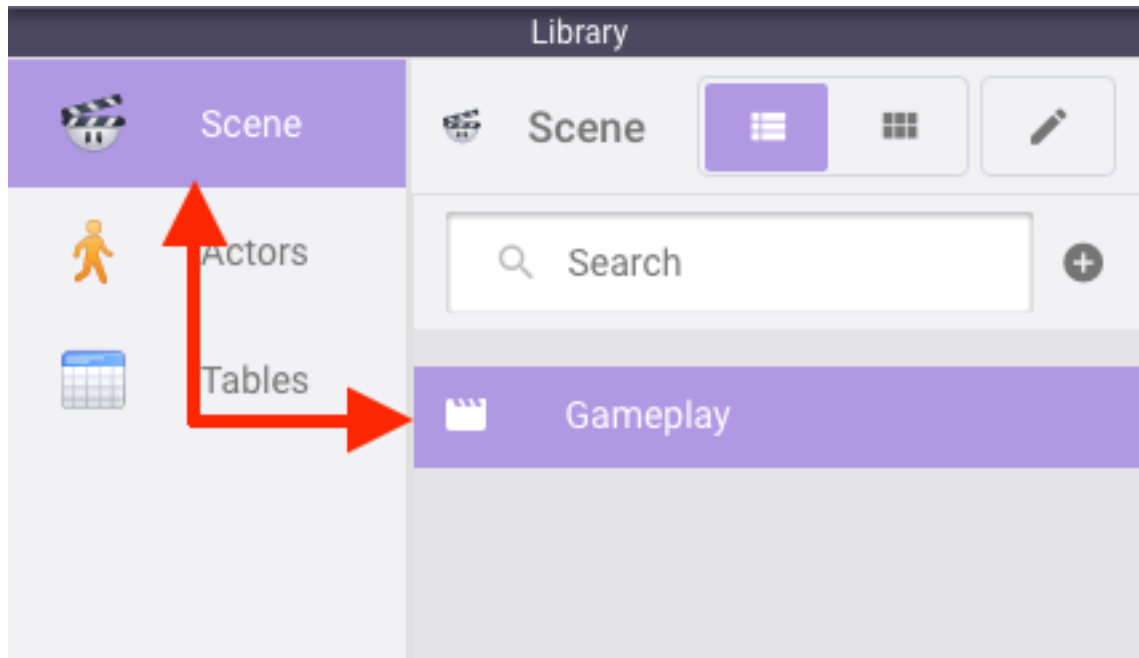
33. Click the Images tab in the Library and locate the “paddleRight” image.
34. Drag the image on top of the white square in the top right of the Inspector. Release the mouse button to apply the image to the actor.



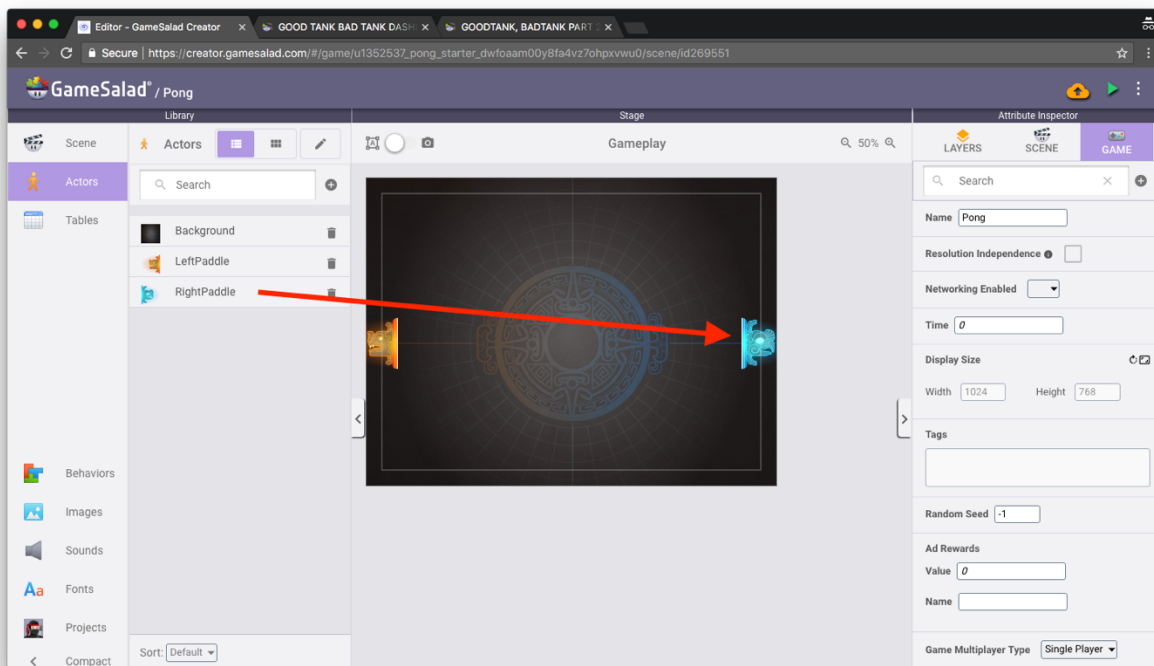
35. Locate the 'Size' Attributes in the Inspector and make sure the 'Width' is '128', and the 'Height' is '128'.



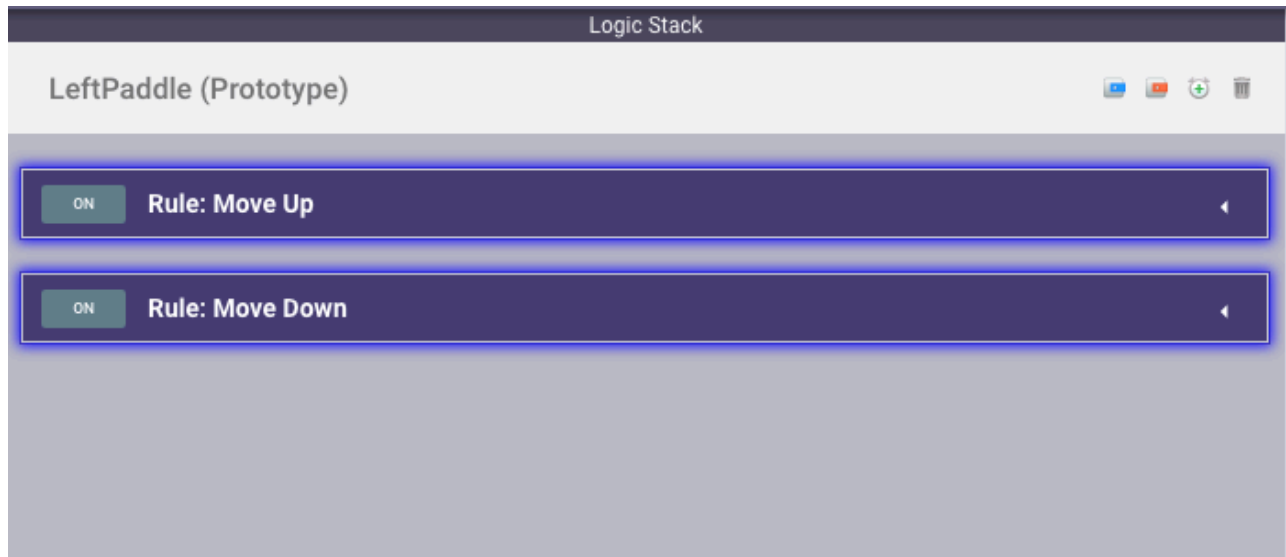
36. Navigate to the Scene Editor for 'Gameplay' by clicking the Scenes tab in the Library and then clicking on the 'Gameplay' Scene.



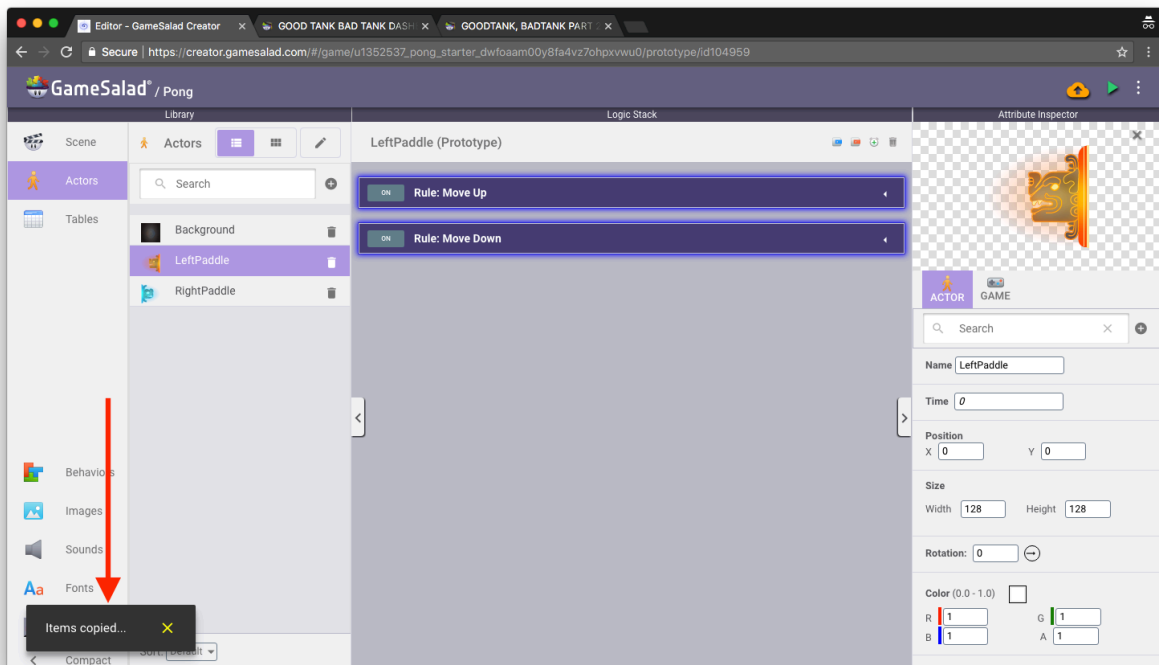
37. Drag the RightPaddle Actor onto the Scene and place it on the right side so that it's opposite the LeftPaddle.



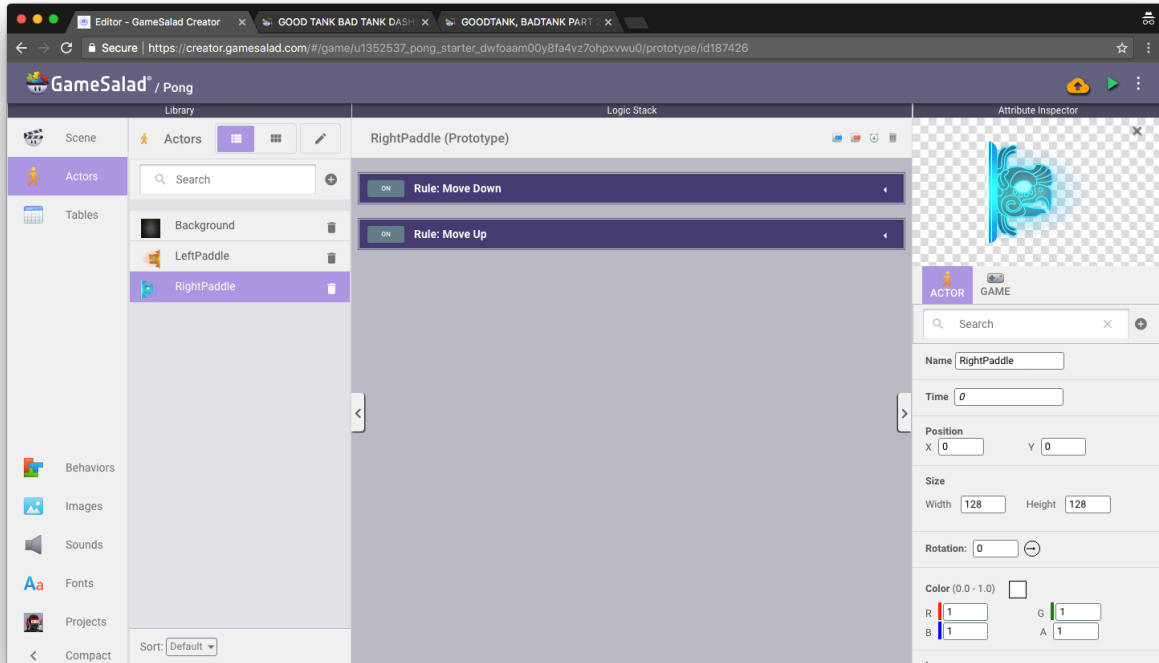
38. Click on the Actors tab in the Library and select the LeftPaddle to open the Actor Editor for it.
39. Select the Move Up and Move Down rules (by left clicking them while holding the shift key down).



40. Use the keyboard shortcut to copy them (control + c if you're on Windows or command + c if you're on Mac). You should see a notification in the bottom left telling you that the behaviors have been copied.



41. Click on the Actors tab in the Library and select the RightPaddle to open the Actor Editor for it.
42. Use the keyboard shortcut to paste the behaviors you just copied (control + v if you're on Windows or command + v if you're on Mac).
43. Now the RightPaddle Actor should have Move Up and Move Down rules.

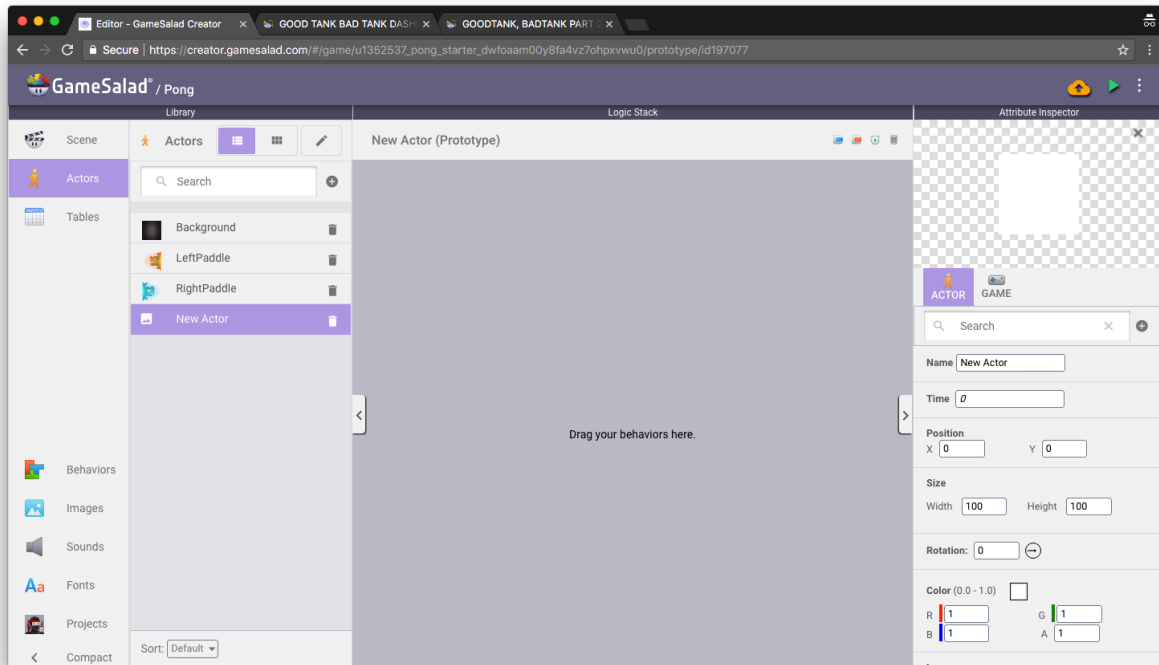


44. Preview the game to make sure that both paddles move up and down when you press the up and down arrow keys.

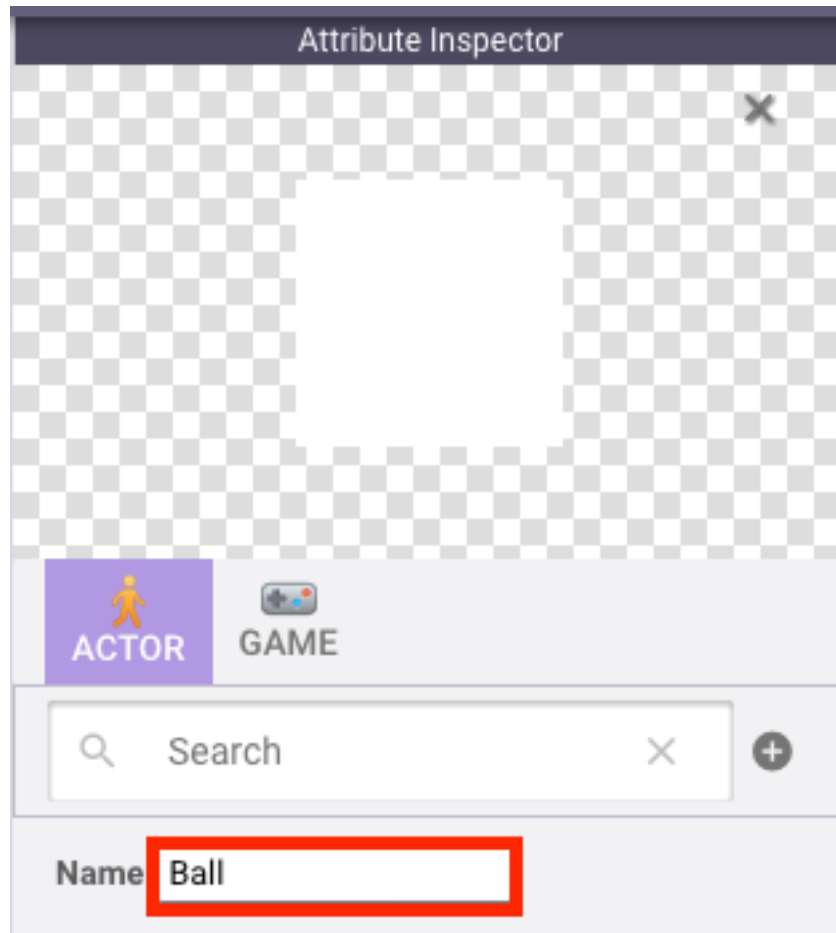
Creating the Ball and Adding Collisions

Now that we have the paddles working, let's add the ball.

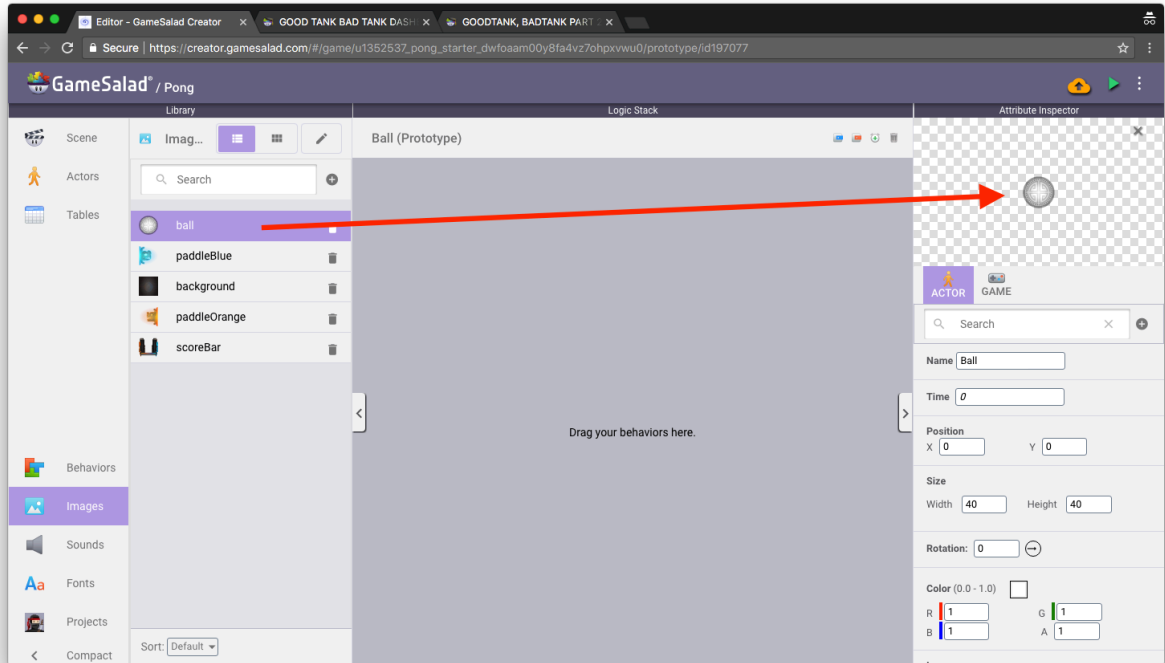
1. Click the Actors tab in the Library and click the '+' icon to the right of the Search Bar to create a new actor.
2. Click on the newly created actor to navigate to the Actor Editor for this specific actor.



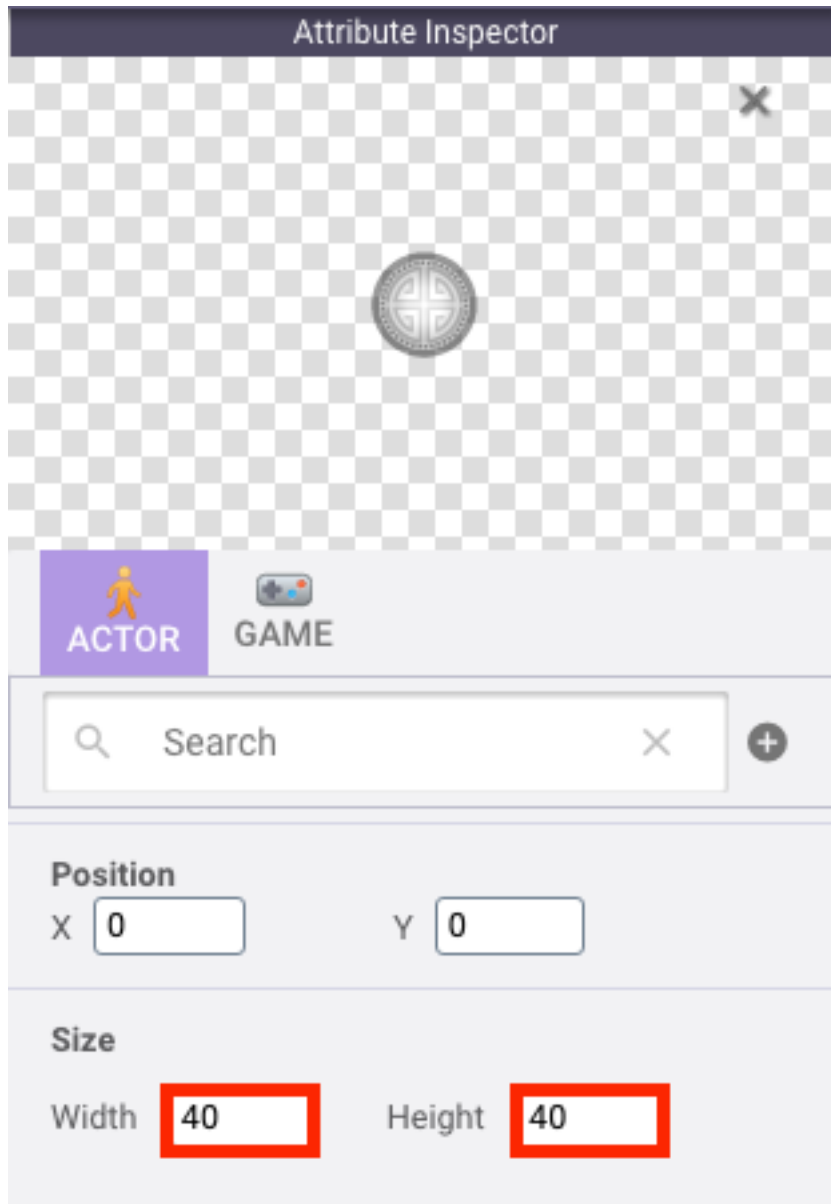
3. In the Inspector on the right hand side, locate the 'Name' Attribute and change the name of the actor to 'Ball'.



4. Click the Images tab in the Library and locate the “ball” image.
5. Drag the image on top of the white square in the top right of the Inspector. Release the mouse button to apply the image to the actor.

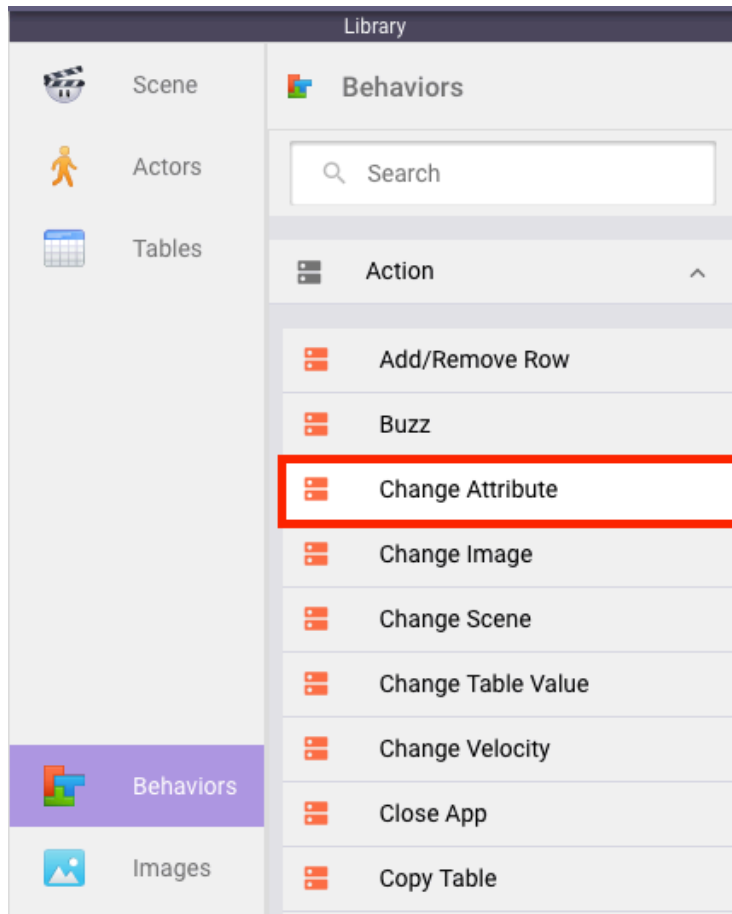


6. Locate the 'Size' Attributes in the Inspector and make sure the 'Width' is '40', and the 'Height' is '40'.

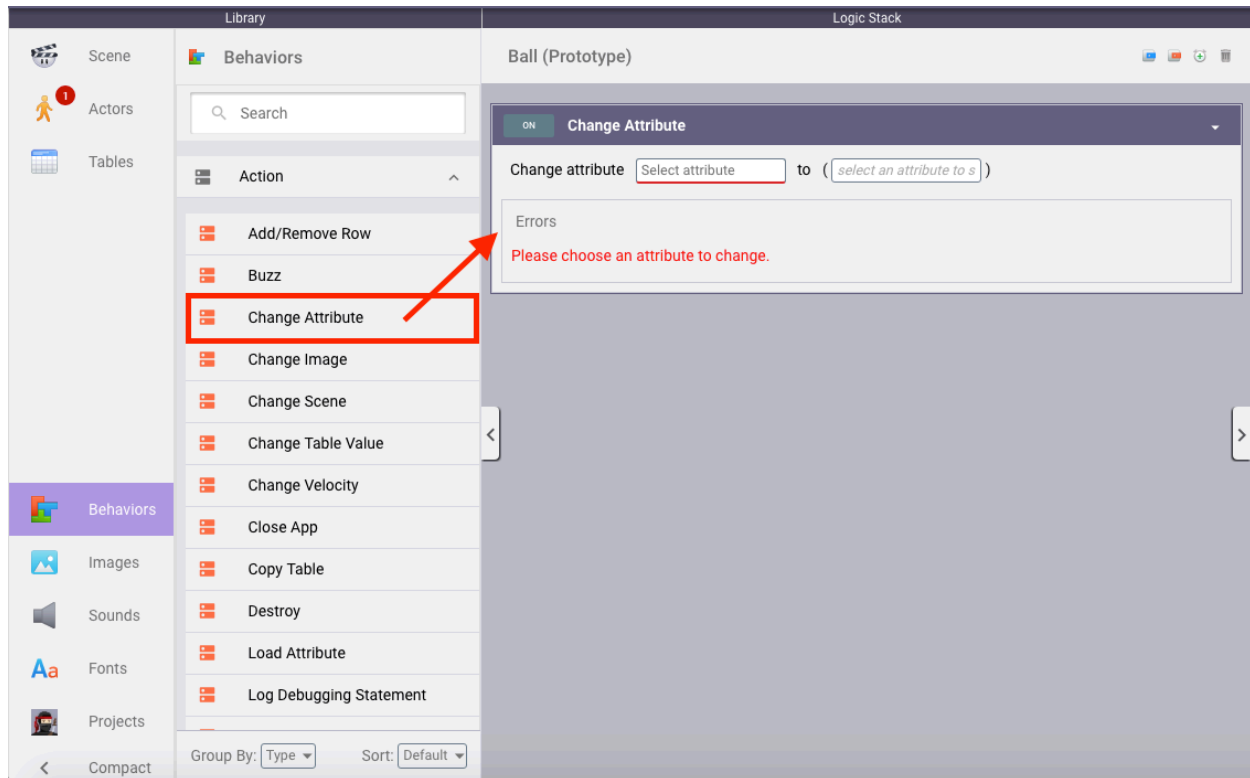


By default, the ball isn't going to do anything, so let's make it move at the start of the game!

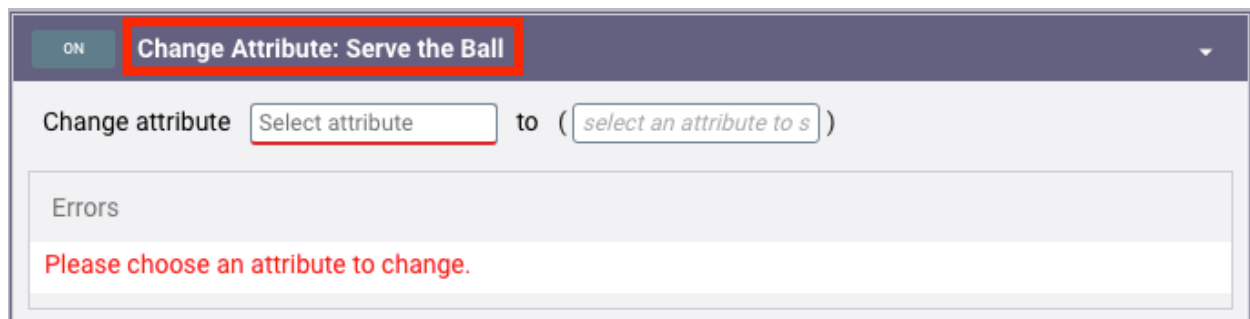
7. Click on the Behaviors tab in the Library and locate the change attribute Behavior.



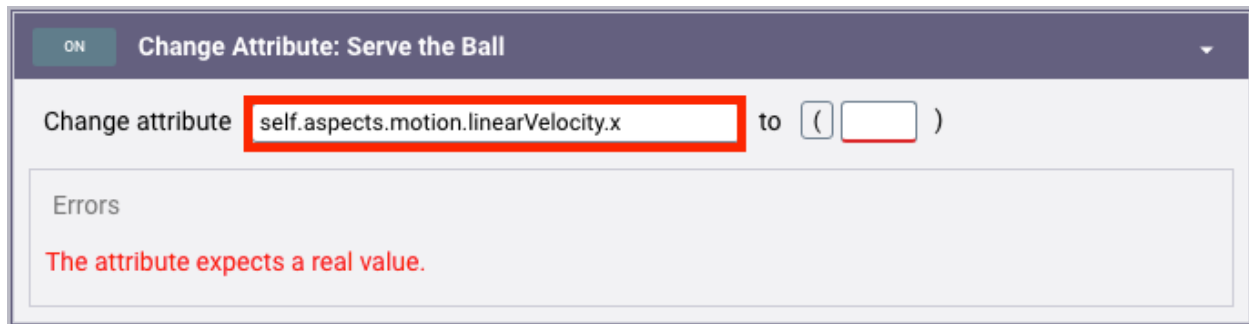
8. Drag the change attribute Behavior into the Logic Stack for the Ball actor.



9. Double click on the title of the behavior to rename it. Rename it 'Change Attribute: Serve the Ball'.



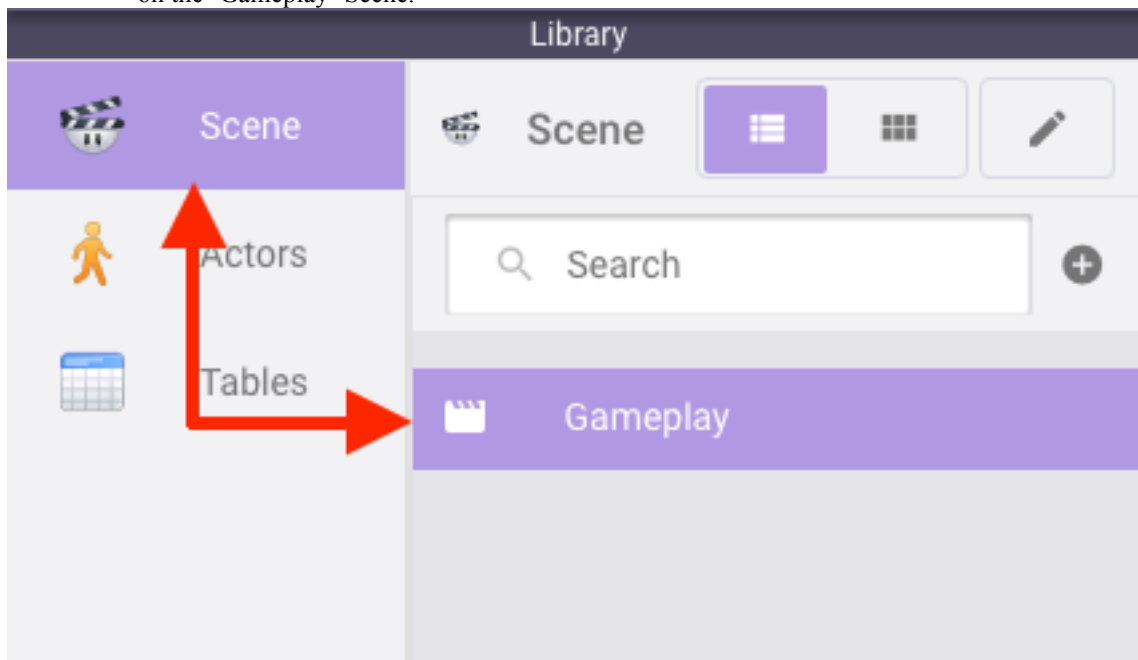
10. Click in the blank field labeled 'select attribute' and choose self > motion > linear velocity > x. (this is the attribute that controls the left/right movement of the actor)



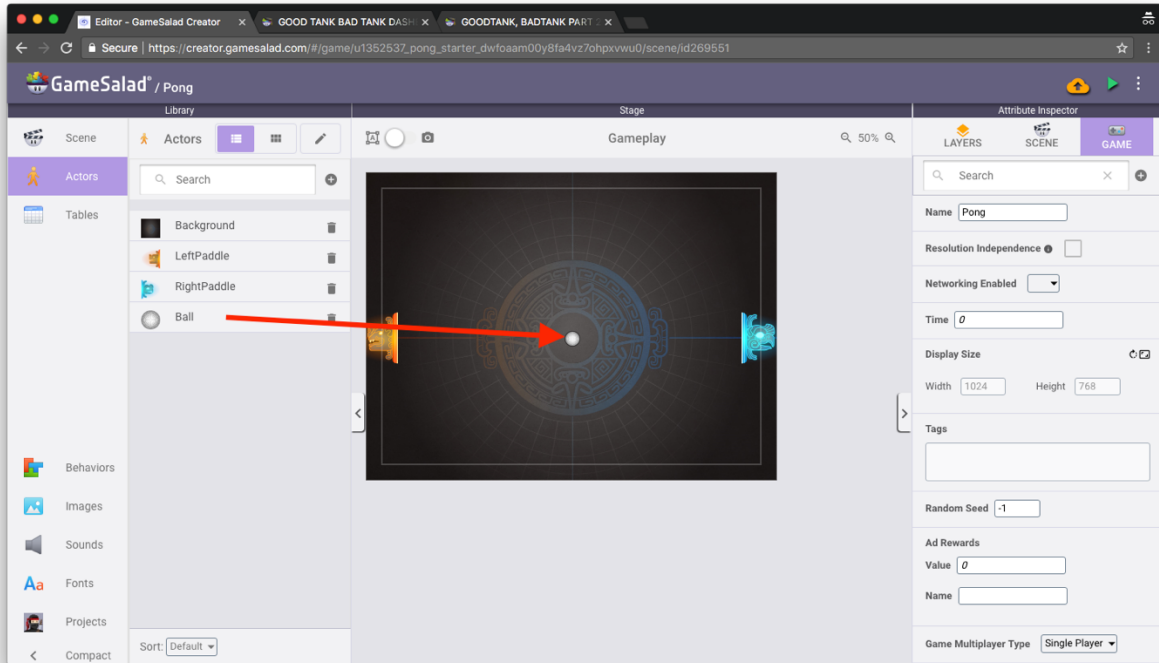
11. In the blank 'to' field, type 300. This behavior will cause the Ball Actor to set its linear velocity x value to 300 at the start of the game, which will cause it to move directly to the right at a speed of 300.



12. Navigate to the Scene Editor for 'Gameplay' by clicking the Scenes tab in the Library and then clicking on the 'Gameplay' Scene.



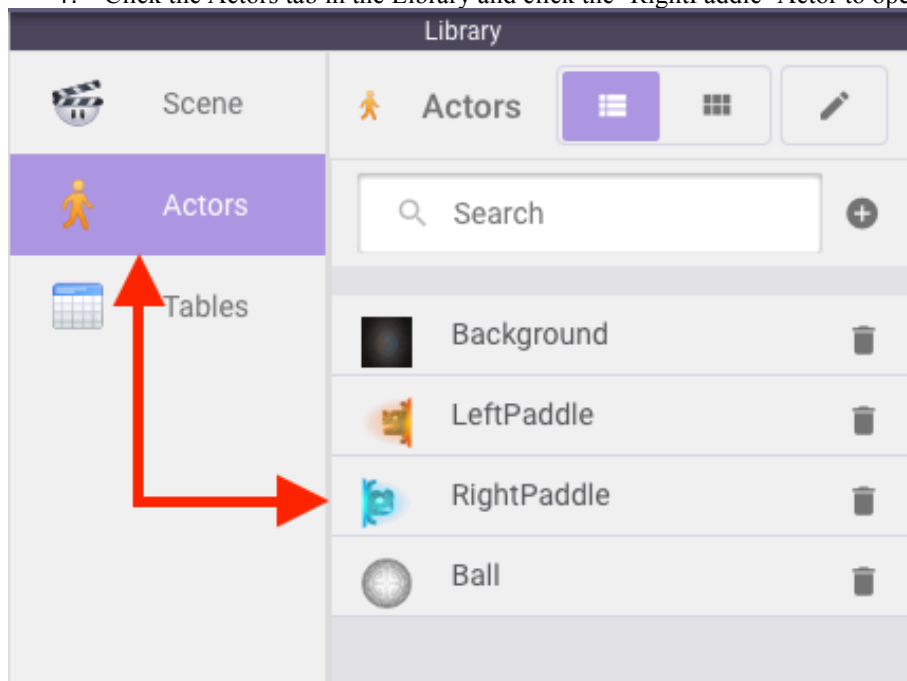
13. Select the Actors tab in the Library and locate the Ball Actor.
14. Drag the Ball Actor onto the scene between the two paddles.



Click the green Preview button in the upper right hand corner of the Scene Editor to Preview the game and see how the actors behave.


You should notice that the Ball starts moving to the right immediately, but passes right through the paddle. Let's fix that!


1. Click the Actors tab in the Library and click the 'RightPaddle' Actor to open the Actor Editor for it.





2. Click on the Behaviors tab in the Library and locate the collide Behavior.


Library


Scene


Actors


Tables


Behaviors

Images

Sounds

Fonts

Projects

Compact

Behaviors

Search

Rule

Timer

Persistent

Accelerate

Accelerate Toward

Animate

Change Size

Collide

Constrain Attribute

Control Camera

Display Text

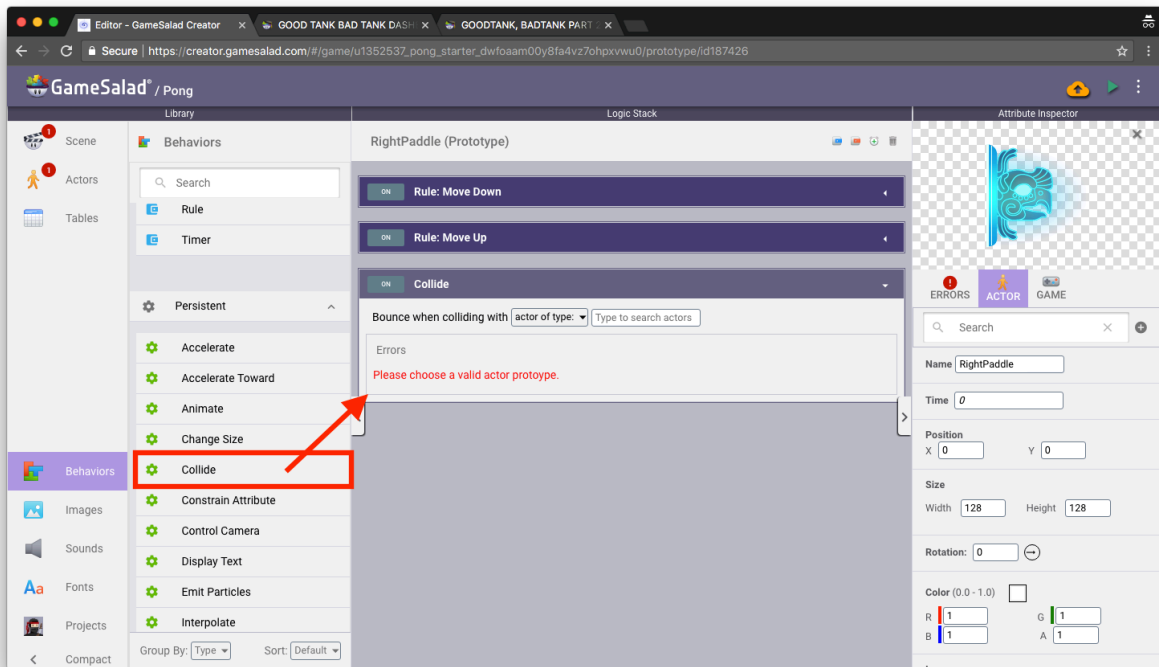
Emit Particles

Interpolate

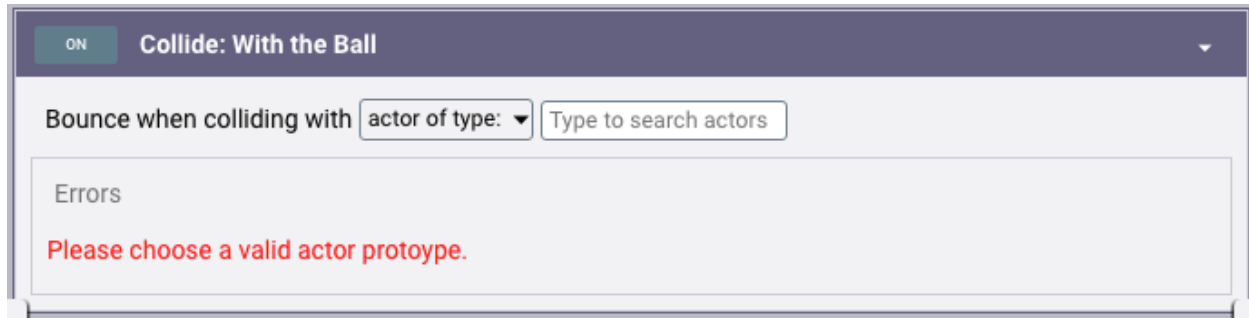
Group By: Type

Sort: Default

3. Drag the collide Behavior into the Logic Stack.



4. Rename the collide behavior 'Collide: With the Ball'

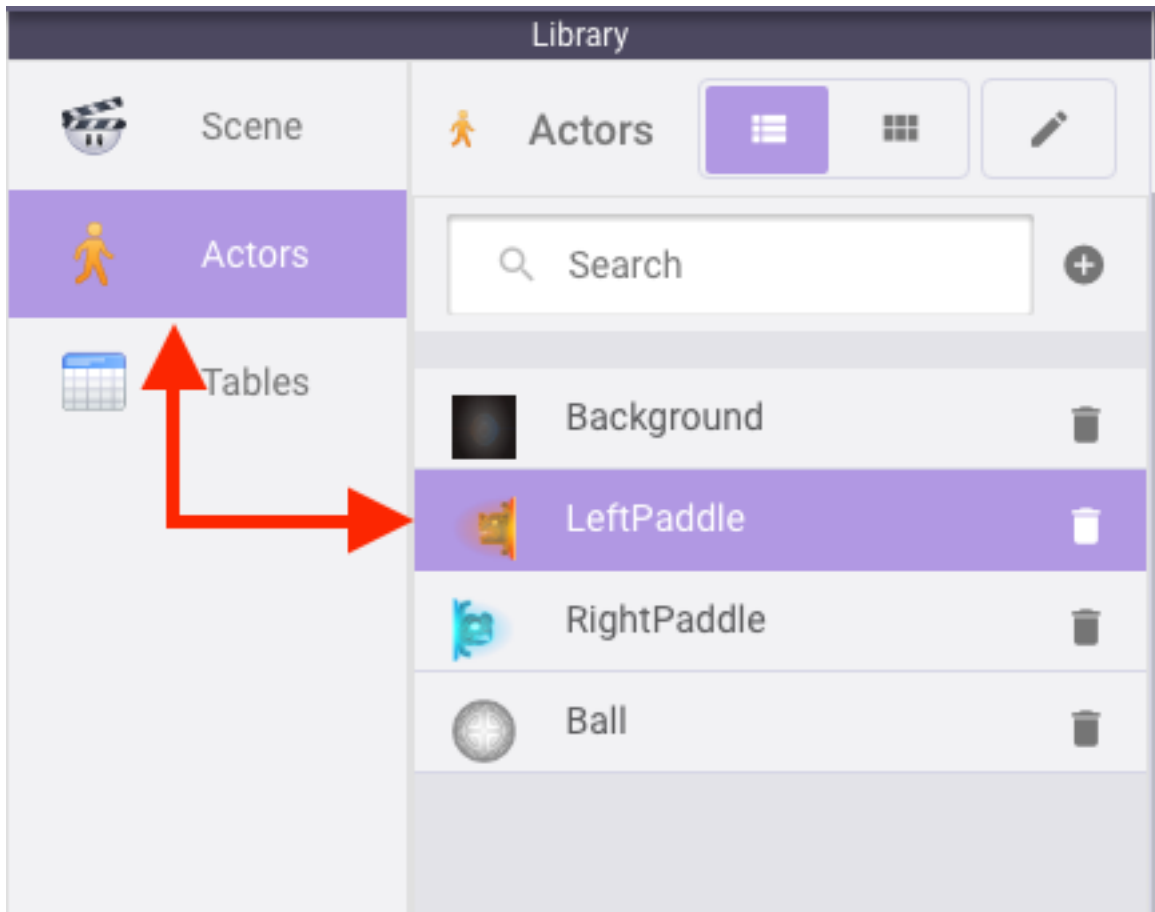


5. Click in the blank field of the collide Behavior and select the Ball actor.

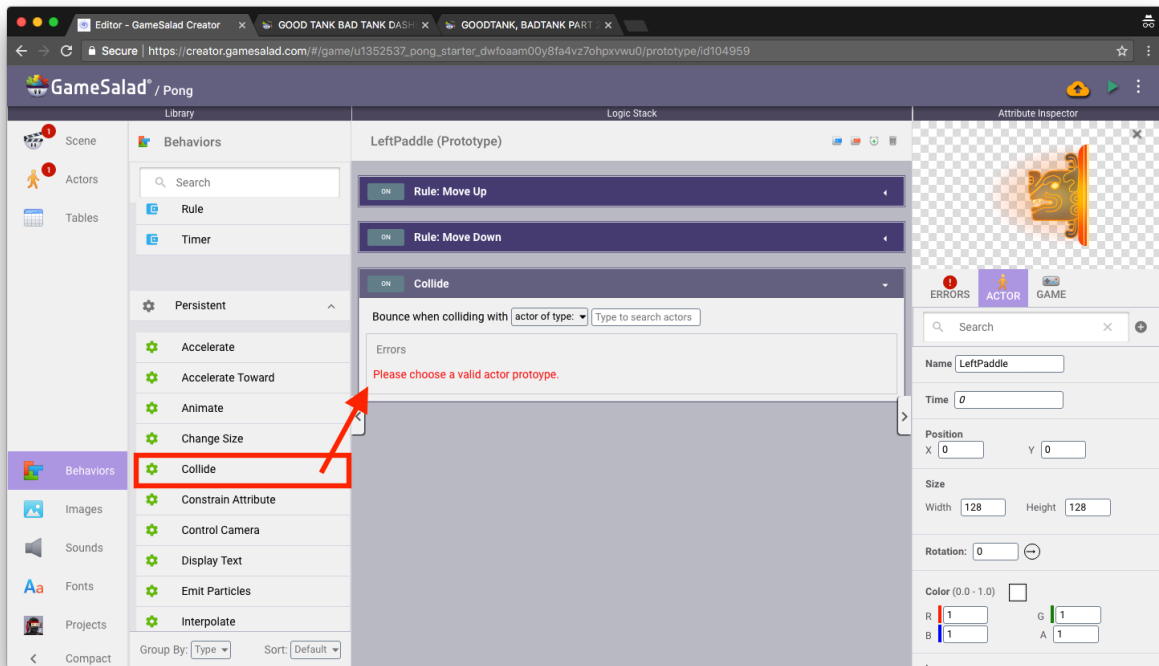


We need to make sure we add the same behavior for the LeftPaddle as well!

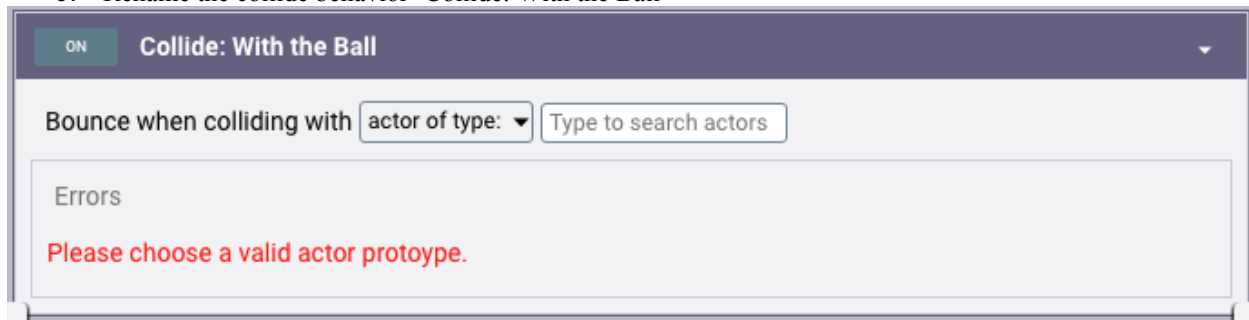
6. Click the Actors tab in the Library and click the 'LeftPaddle' Actor to open the Actor Editor for it.



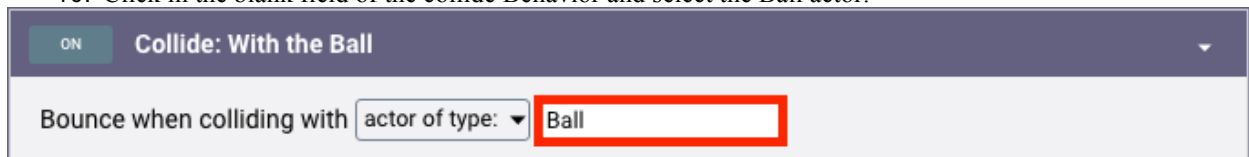
7. Click on the Behaviors tab in the Library and locate the collide Behavior.
8. Drag the collide Behavior into the Logic Stack.



9. Rename the collide behavior 'Collide: With the Ball'



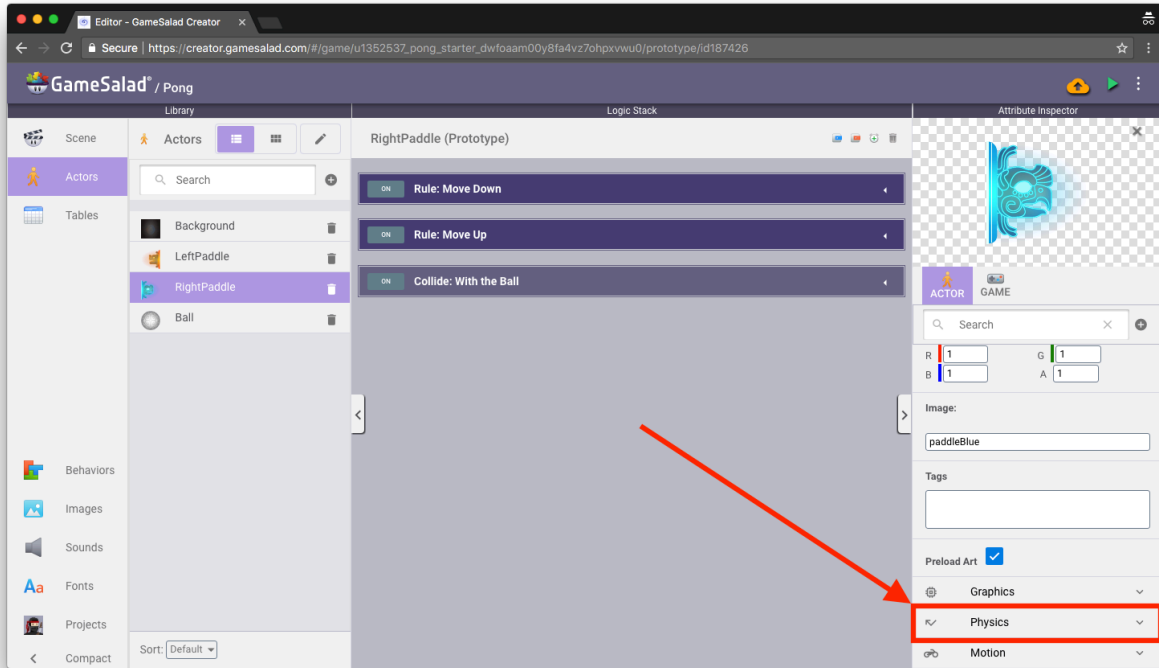
10. Click in the blank field of the collide Behavior and select the Ball actor.



Preview the game again to see how it behaves now with the collisions added.

You should see the ball correctly collide with the paddles, but in doing so, it pushes the paddles off the screen and makes them rotate. We can fix these results through editing some physics attributes for the paddles and ball actors.

11. Click on the Actors tab in the Library and click on the RightPaddle Actor to open the Actor Editor.
12. In the Inspector on the right side of the screen, locate the section labeled 'Phycis' near the bottom.



13. Click on the Physics section to expand the list of Physics attributes for this actor.

Attribute Inspector



ACTOR



GAME



Search



Physics



Density

1

Friction

3

Bounciness

1

Fixed Rotation



Movable



Collision Shape

Rectangle ▼

Drag

0

Angular Drag

0

14. Change the Density attribute from '1' to '1000'. This will make this actor act like it weighs a lot more than the ball. *(maybe talk about the definition of density here)*
15. Change the Friction attribute from '3' to '0'. This will make sure that the ball doesn't slow down after bouncing off the paddle. *(maybe talk about the definition of friction here)*
16. Click in the box next to the Fixed Rotation attribute to turn it on.

Attribute Inspector



ACTOR

GAME



Search



Physics



Density

1000

Friction

0

Bounciness

1

Fixed Rotation



Movable

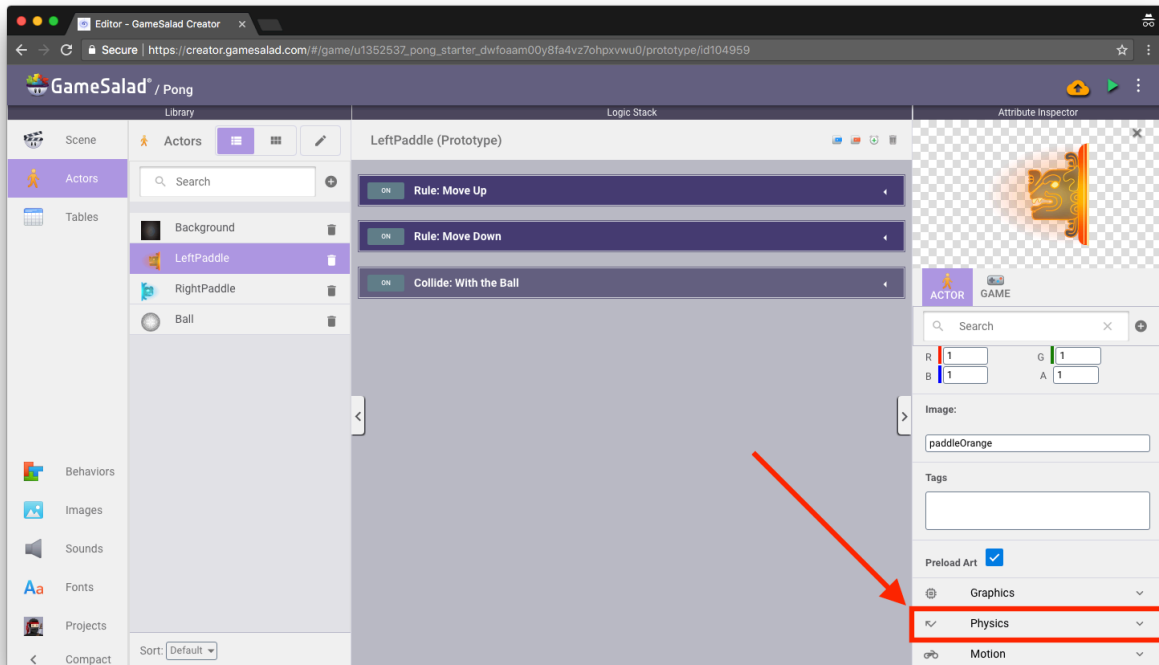


Collision Shape

Rectangle ▼

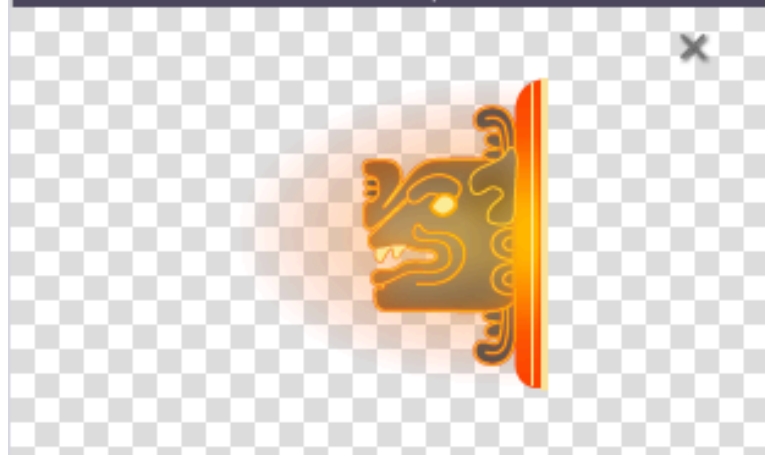
Preview the game again to make sure that the ball can no longer push the right paddle when they collide. Let's make sure to update these same attributes in the LeftPaddle Actor too so that it behaves the same way.

17. Click on the Actors tab in the Library and click on the LeftPaddle Actor to open the Actor Editor.
18. In the Inspector on the right side of the screen, locate the section labeled 'Phycsis' near the bottom.



19. Click on the Physics section to expand the list of Physics attributes for this actor.

Attribute Inspector



ACTOR



GAME



Search



Physics



Density

1

Friction

3

Bounciness

1

Fixed Rotation

☐

Movable



Collision Shape

Rectangle ▾

Drag

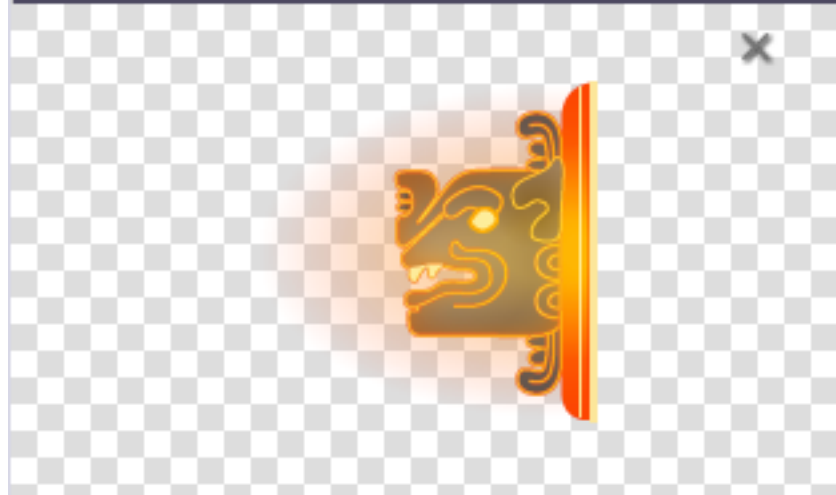
0

Angular Drag

0

20. Change the Density attribute from '1' to '1000'. This will make this actor act like it weighs a lot more than the ball.
21. Change the Friction attribute from '3' to '0'. This will make sure that the ball doesn't slow down after bouncing off the paddle.
22. Click in the box next to the Fixed Rotation attribute to turn it on.

Attribute Inspector



ACTOR

GAME



Search



Physics



Density

1000

Friction

0

Bounciness

1

Fixed Rotation



Movable



Collision Shape

Rectangle



Preview the game again to make sure that the ball doesn't push either of the paddles off the screen.

There's a couple more physics attributes we should edit in the Ball Actor as well before we move on.

23. Click on the Actors tab in the Library and click on the Ball Actor to open the Actor Editor.
24. In the Inspector on the right side of the screen, locate the section labeled 'Phycis' near the bottom.
25. Click on the Physics section to expand the list of Physics attributes for this actor.

Attribute Inspector



ACTOR



GAME



Search



Physics



Density

1

Friction

3

Bounciness

1

Fixed Rotation

☐

Movable



Collision Shape

Rectangle ▼

Drag

0

Angular Drag

0

26. Change the Friction attribute from '3' to '0'. This will make sure that the ball doesn't slow down after bouncing off the paddle.
27. Click on the Collision Shape dropdown box, and change it to 'Circle'. This will cause the Ball actor to act like a circle when it collides with other actors as opposed to acting like a square.

Attribute Inspector



ACTOR



GAME



Search



Physics



Density

1

Friction

0

Bounciness

1

Fixed Rotation



Movable



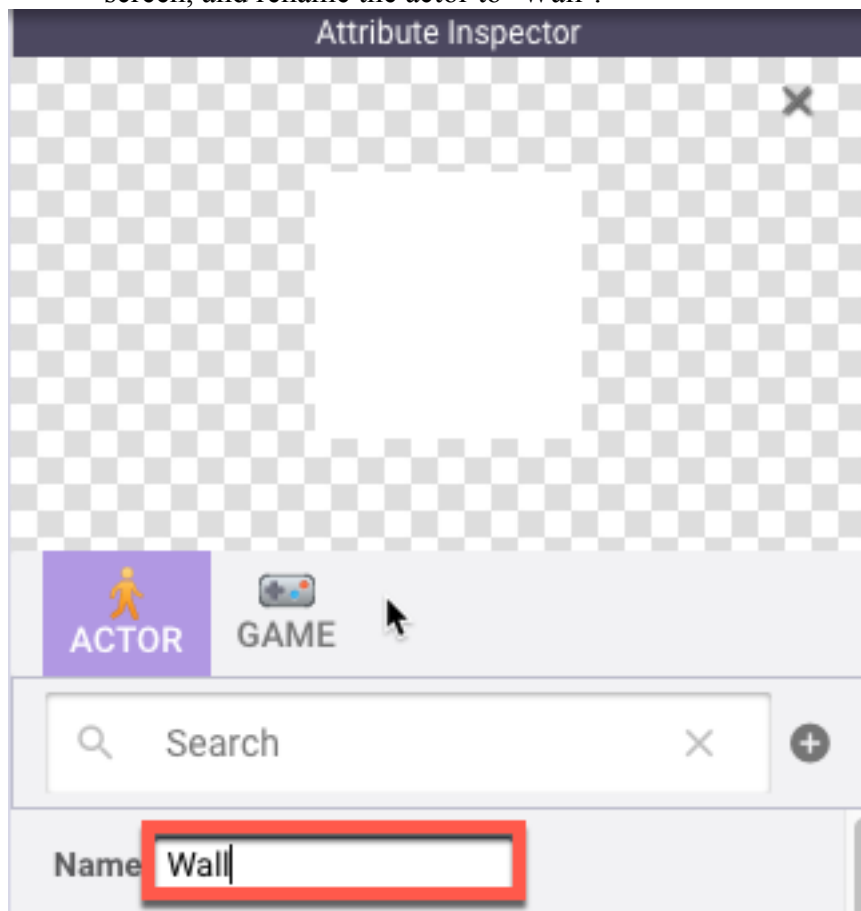
Collision Shape

Circle ▼

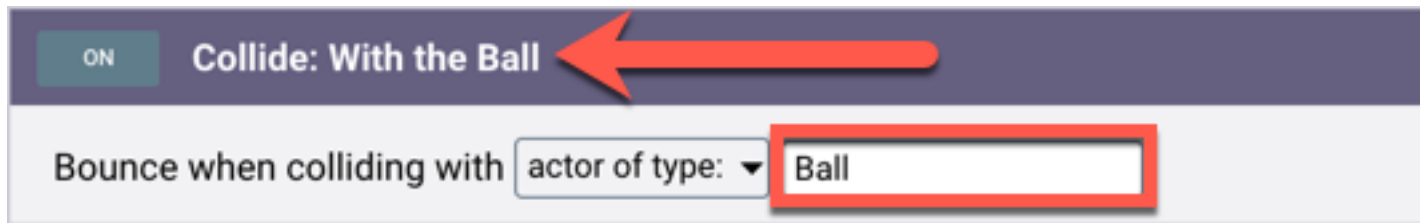
Preview the game again to make sure everything is behaving correctly. You may have noticed if you hit the ball up or down at all that it just flies off the screen instead of staying where we can see it.

Let's add a couple walls to fix that.

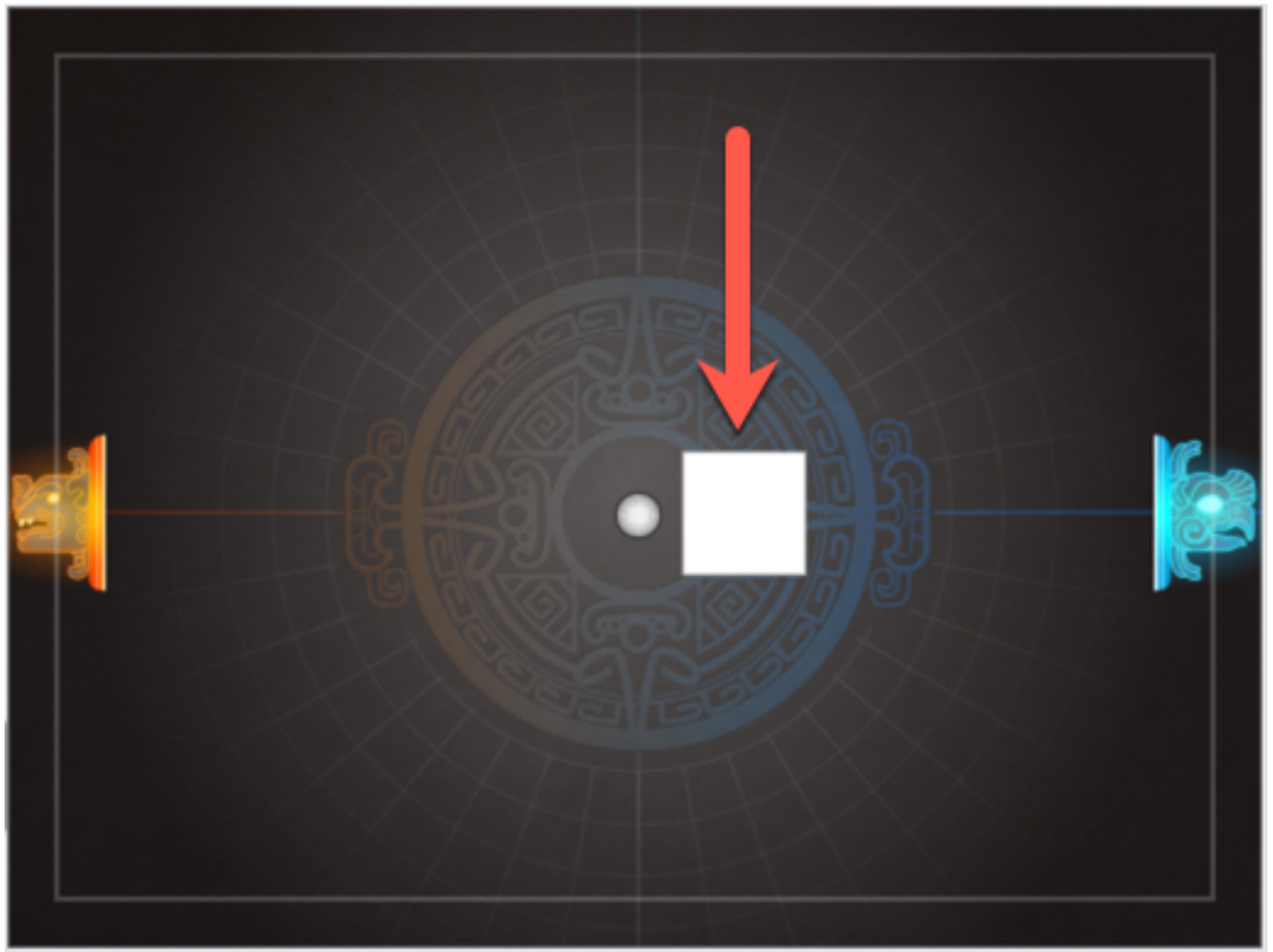
28. Click on the Actors tab in the Library and click the '+' button next to the search bar to create a new Actor.
29. Click on the newly created actor to open the Actor Editor.
30. Locate the 'name' attribute for the actor in the inspector over on the right side of the screen, and rename the actor to 'Wall'.



31. Click on the behaviors tab in the Library and locate the collide behavior.
32. Drag the collide behavior into the logic stack to add it to the wall actor
33. Rename the behavior to "Collide: With the Ball"
34. Click in the blank field of the collide behavior and select the Ball actor from the list that appears.



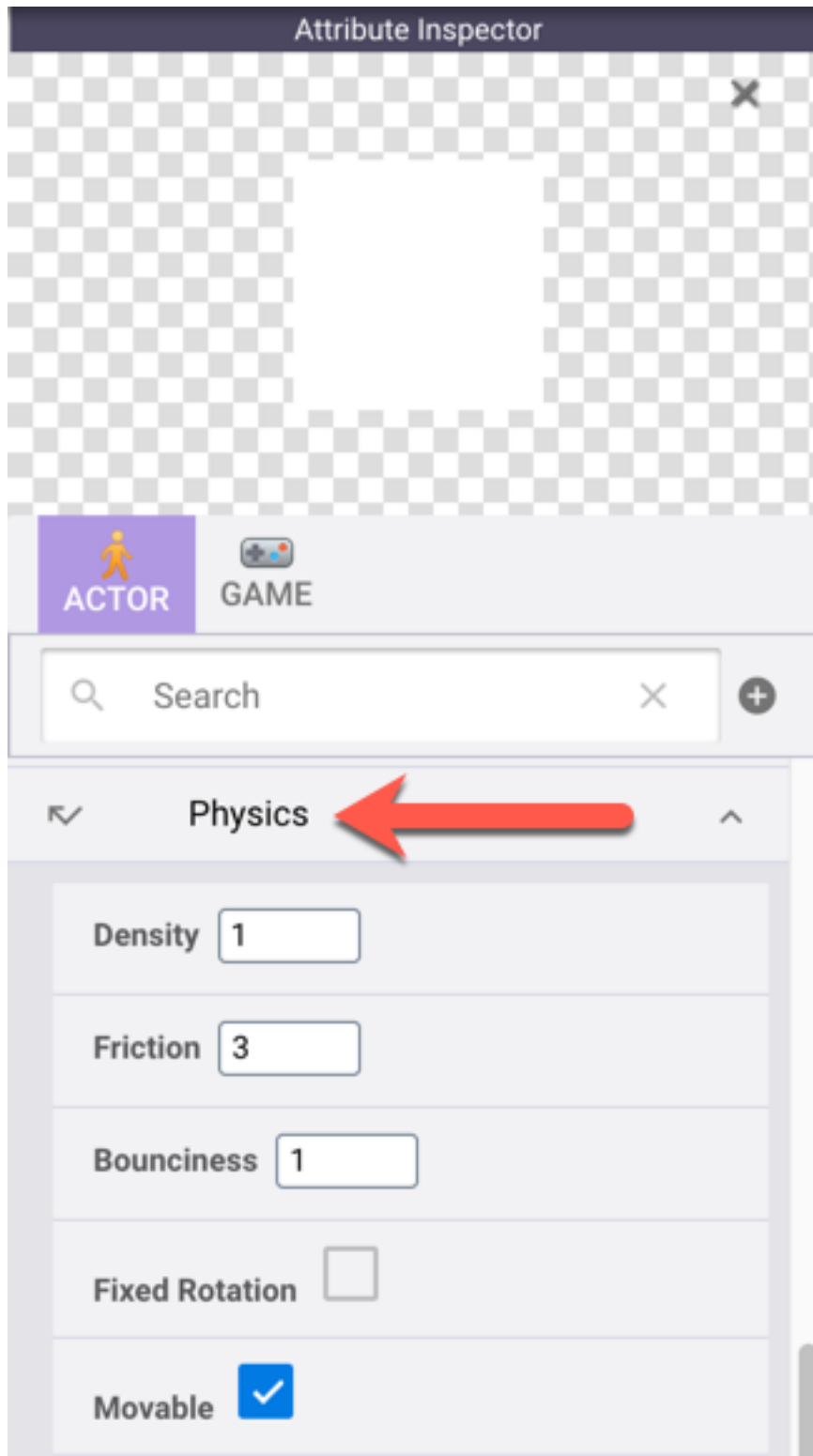
35. Click on the scene tab in the library and select the Gameplay scene to open the Scene Editor.
36. Drag an instance of the wall actor onto the scene and position it to the right of the ball actor. This is so we can see how they interact with each other when they collide.



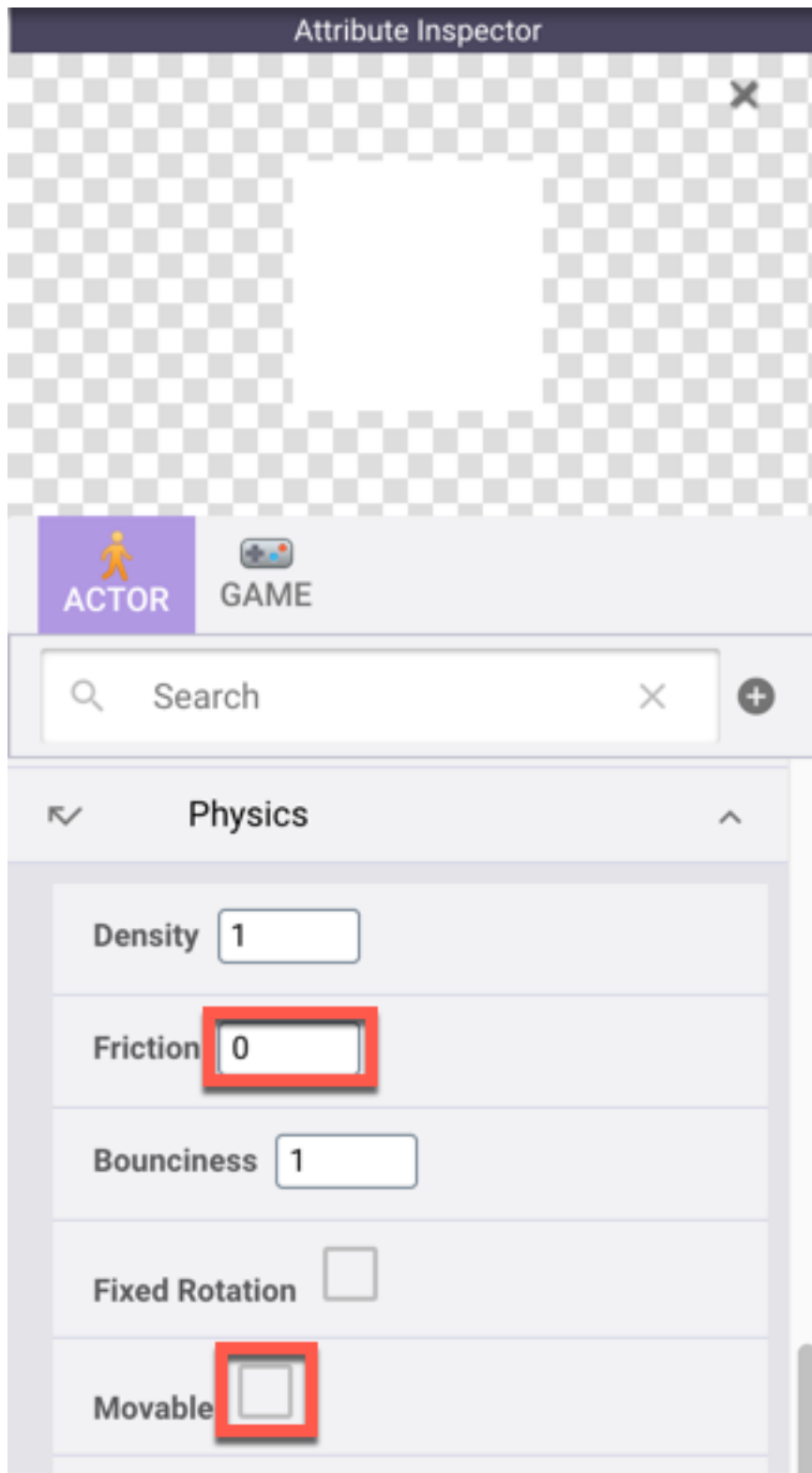
Preview the game to see if the wall is working how we want it to. You should notice that the wall gets pushed back by the ball like our paddles were earlier. Let's fix this through the physics attributes in the wall actor.

37. Click on the actors tab in the Library and select the wall actor.

38. In the inspector on the right side of the screen scroll down and locate the list of physics attributes for the actor.
39. Click on the list of physics attributes to expand it.

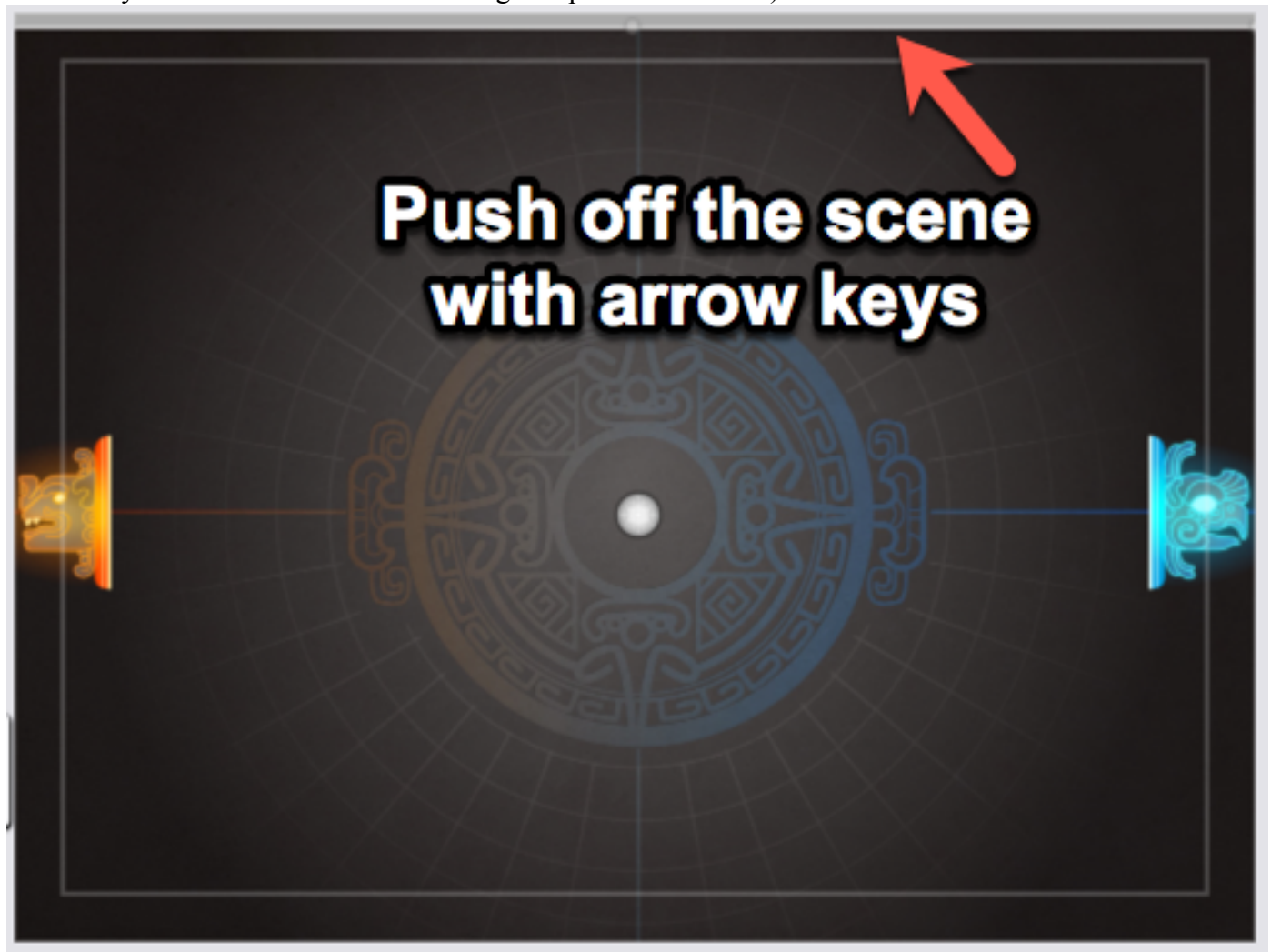


40. Change the friction attribute to '0'.
41. Click the box next to the movable attribute to turn it off (since we don't want our walls to move at all).



Preview the game again to make sure the wall is working as expected. Now that the walls are working correctly, let's position them on the scene.

42. Click on the scene tab in the library and select the Gameplay scene to open the Scene Editor.
43. Resize the wall actor currently on the scene so that it's as wide as the scene and position it just outside the camera on the top of the scene. (Note: you may need to use the arrow keys with the actor selected to nudge it up outside of view).



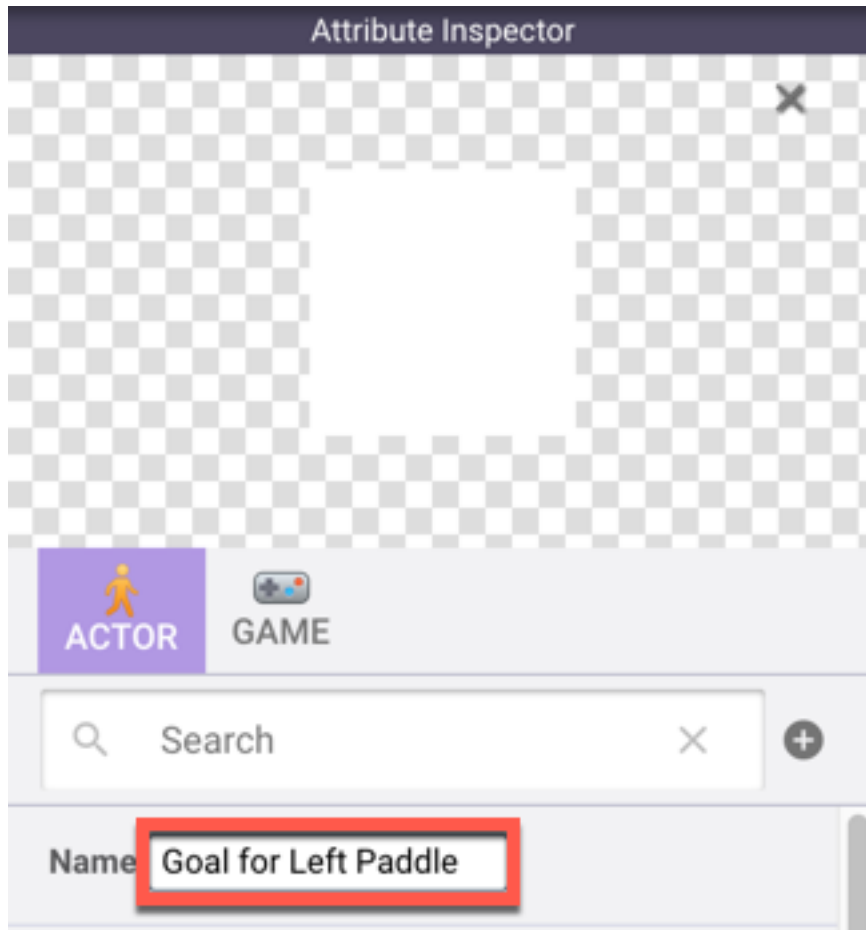
44. Drag another wall actor onto the scene from the actors tab in the library.
45. Resize the new wall so that it's as wide as the scene and position it just outside the camera on the bottom of the scene. (Note: you may need to use the arrow keys with the actor selected to nudge it down outside of view).



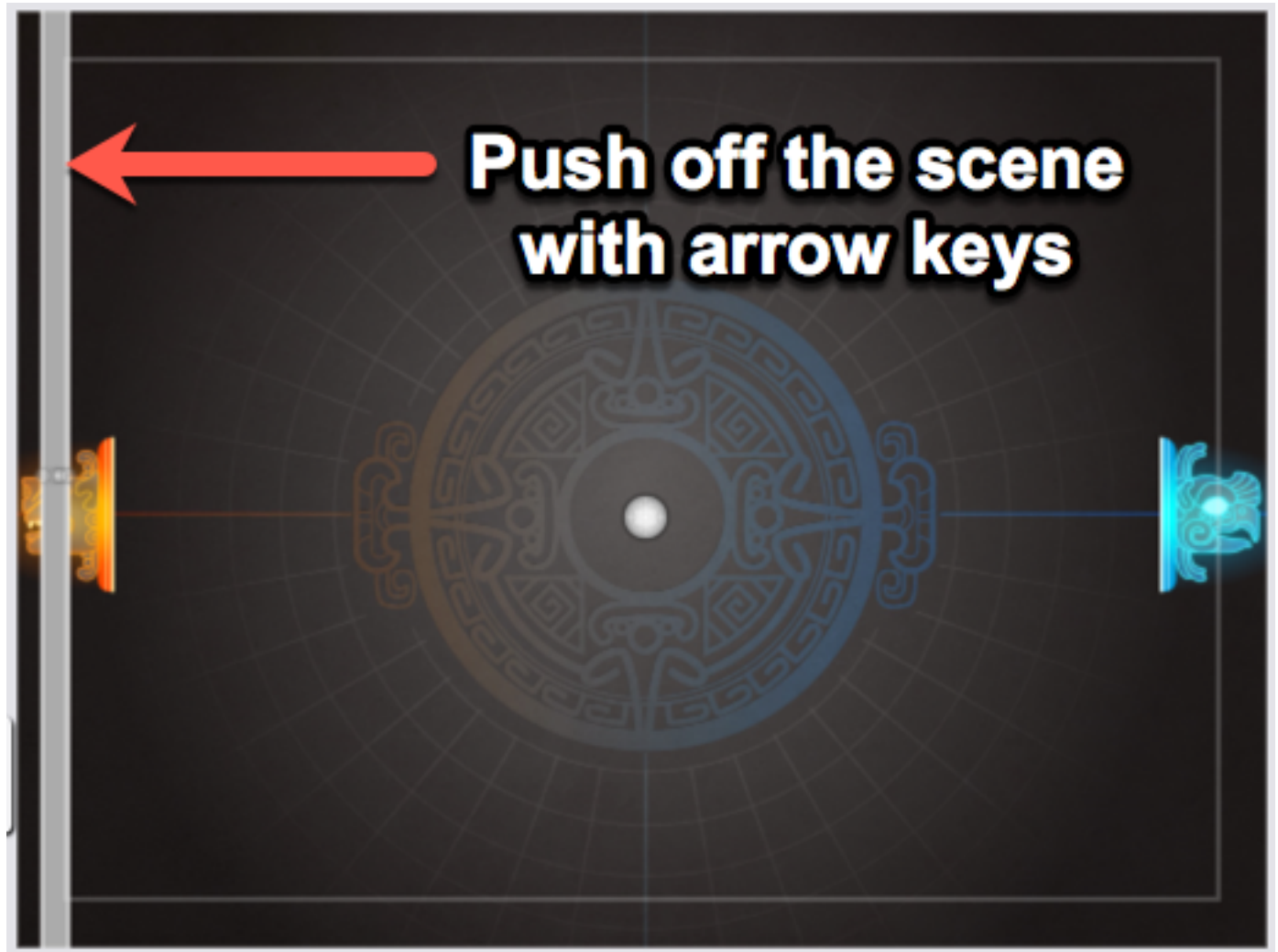
Now that we have walls to stop the ball from going off the top or bottom of the screen, we'll need something to handle when the ball goes past one of the paddles.

Adding a Score System

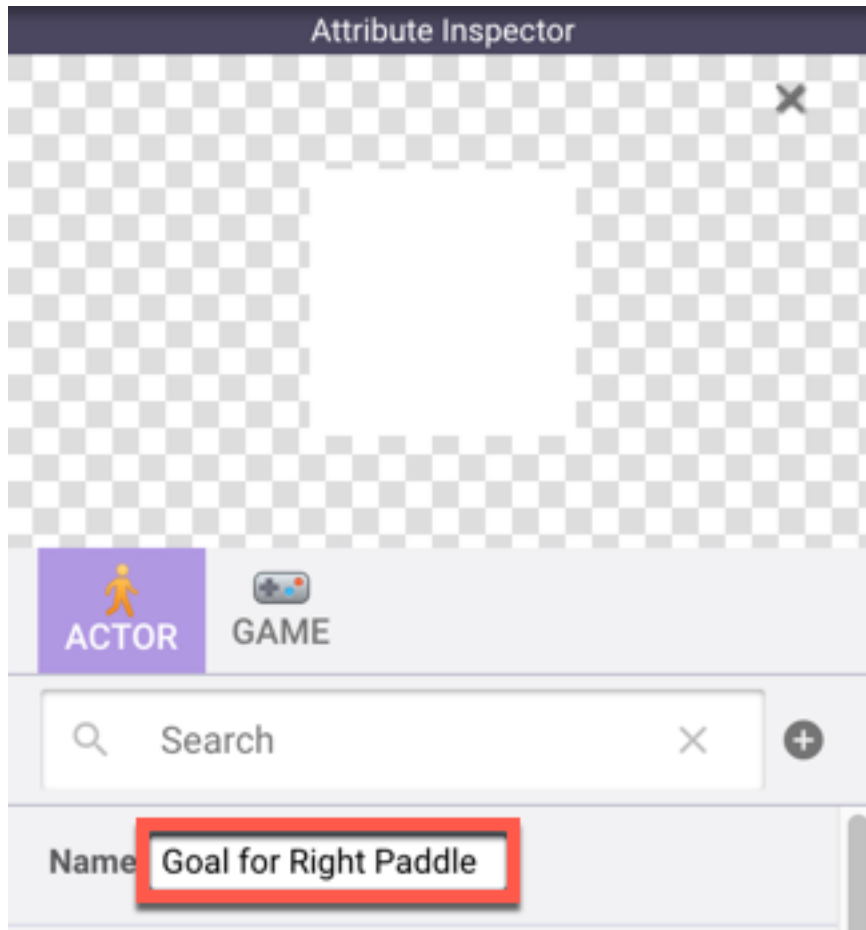
1. Click on the Actors tab in the Library and click the '+' button next to the search bar to create a new Actor.
2. Click on the newly created actor to open the Actor Editor.
3. Locate the 'name' attribute for the actor in the inspector over on the right side of the screen, and rename the actor to 'Goal for Left Paddle'.



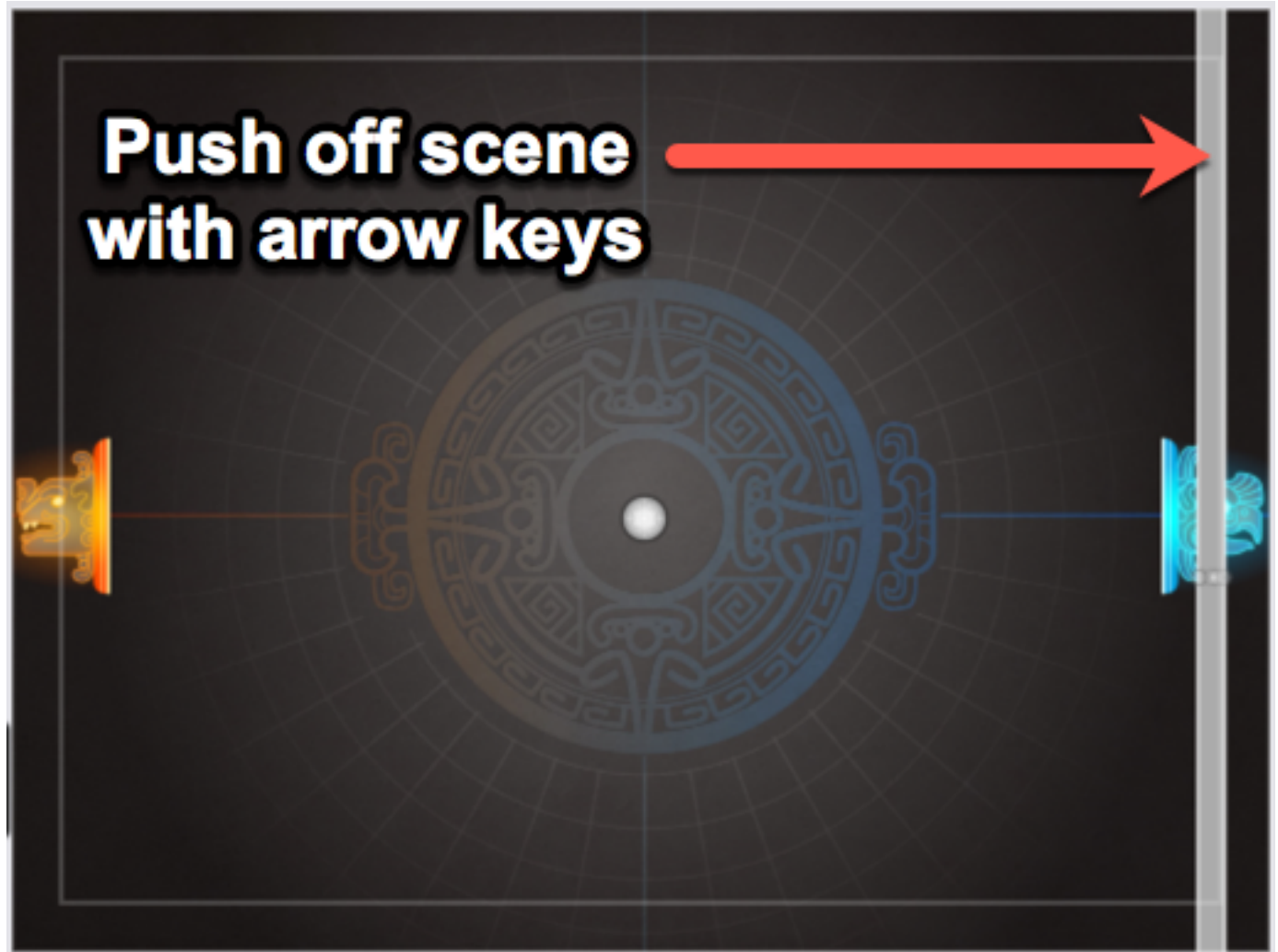
4. Click on the scene tab in the library and select the Gameplay scene to open the Scene Editor.
5. Drag an instance of the Goal for Left Paddle actor onto the scene.
6. Resize it to be the height of the scene and position it just off the screen on the right side. (This actor will serve as the goal that the paddle on the left is aiming for)



7. Click on the Actors tab in the Library and click the '+' button next to the search bar to create a new Actor.
8. Click on the newly created actor to open the Actor Editor.
9. Locate the 'name' attribute for the actor in the inspector over on the right side of the screen, and rename the actor to 'Goal for Right Paddle'.

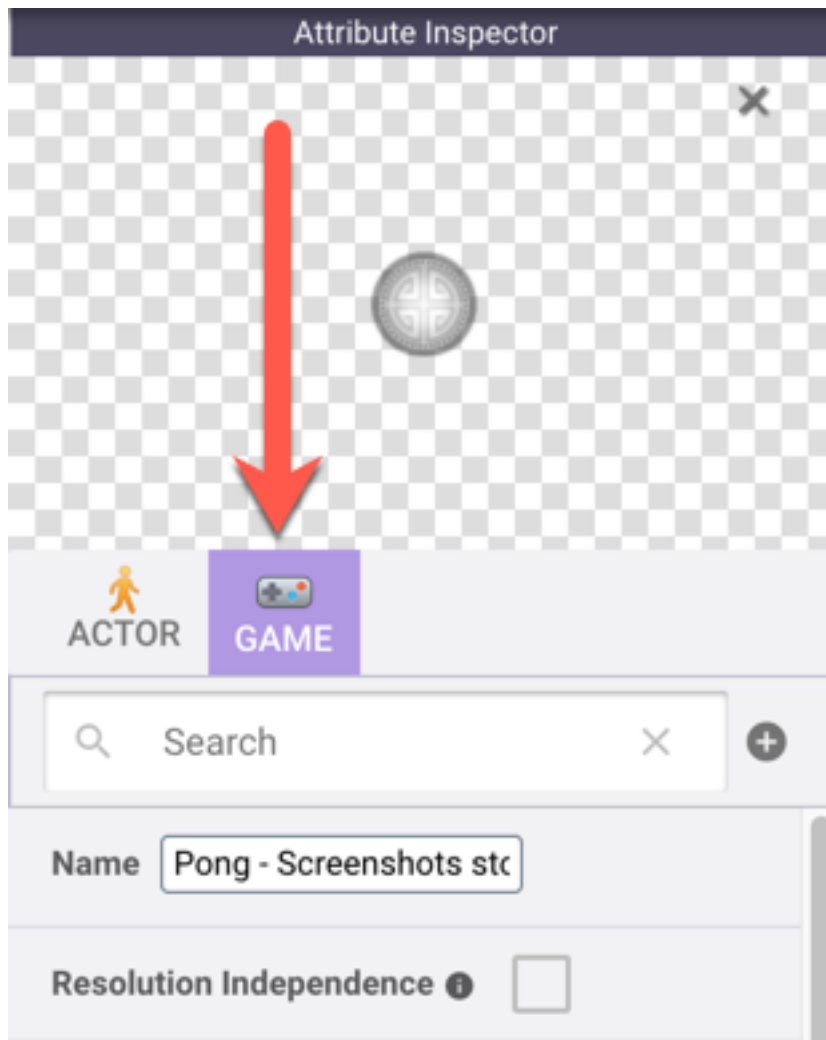


10. Click on the scene tab in the library and select the Gameplay scene to open the Scene Editor.
11. Drag an instance of the Goal for Right Paddle actor onto the scene.
12. Resize it to be the height of the scene and position it just off the screen on the left side.
(This actor will serve as the goal that the paddle on the right is aiming for)

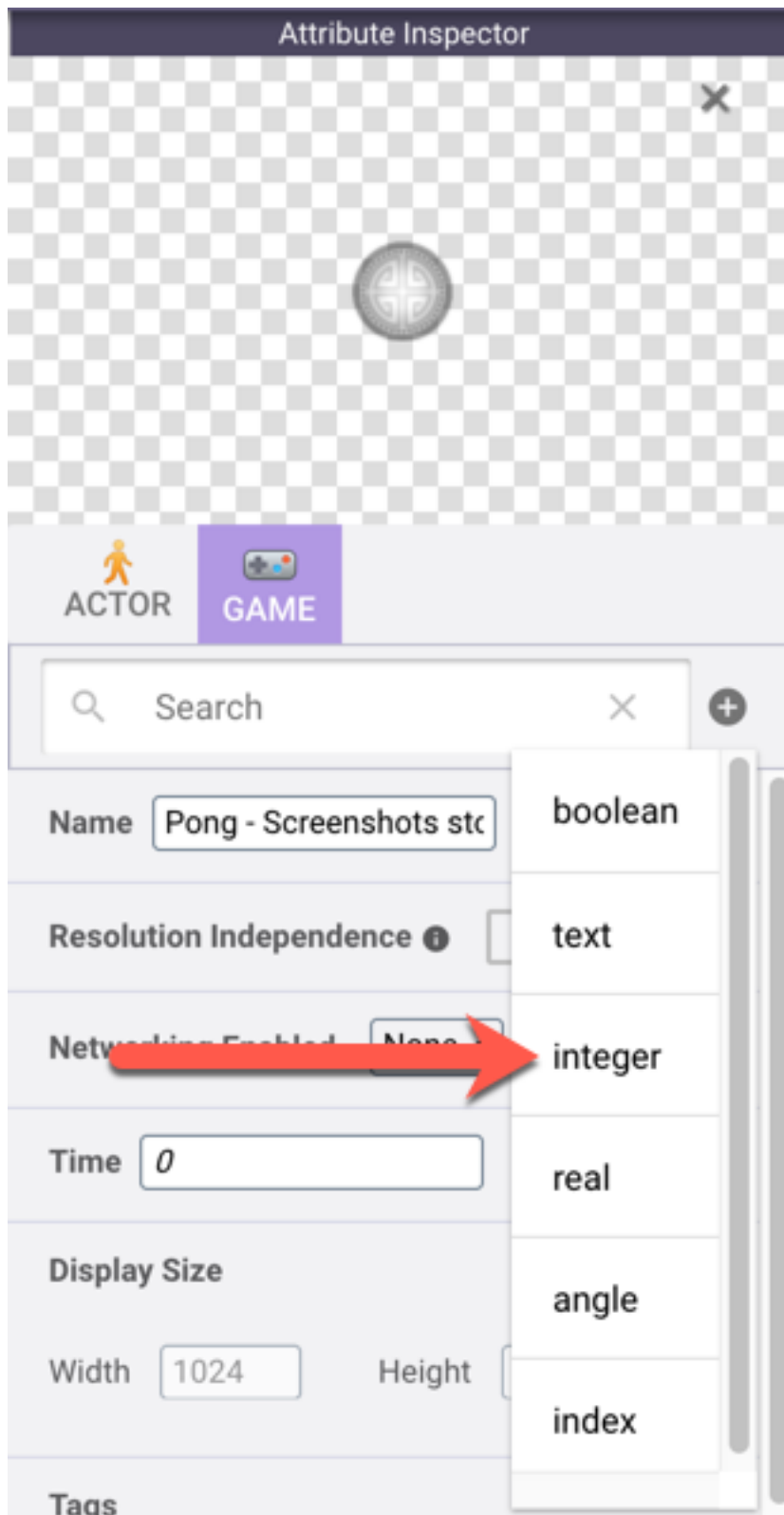


Now that we have our goal actors created, let's go ahead and add the logic to make them work.

13. Click on the actors tab in the library and select the ball actor to open the actor editor.
14. In the inspector on the right side of the screen, click the game tab (this will show you all the current game level attributes that exist in the project).



15. Click the '+' button to the right of the search bar in the inspector to create a new integer attribute.



16. Scroll down to the bottom of the list of attributes to see the new one you just created and double click on the name 'new attribute' to rename it. Rename it to LeftScore.

Game Multiplayer Type Single Player ▾

platformConnected: ☐

LeftScore integer ⊖

17. Click the '+' button again to create another integer attribute, and rename it RightScore.

Game Multiplayer Type Single Player ▾

platformConnected: ☐

LeftScore integer ⊖

RightScore integer ⊖

18. Add a rule to the ball actor by clicking the blue add rule button across the top of the logic stack.

Logic Stack

Ball (Prototype)

Click here

ON Change Attribute: Serve the Ball

Change attribute to ()

ON Rule

If of the following conditions are valid

this actor receives event over actor

🔍 Type down key to select condition or start typing to search.

Then

<

Drop behaviors here

19. Rename the rule to 'Rule: Left Paddle Scored'.

ON

Rule: Left Paddle Scored

If

all

 of the following conditions are valid

this actor receives event

mouse pointer

over actor

🔍

 Type down key to select condition or start typing to search.

Then

Drop behaviors here.

Else

20. Change the condition of the rule from 'mouse pointer' to 'overlaps or collides'.

ON

Rule: Left Paddle Scored

If

all ▼

of the following conditions are valid

this actor receives event

overlaps or collides ▼

actor of type ▼

Type to search actor

🔍

Type down key to select condition or start typing to search.

Then

Drop behaviors here.

Else

21. Click in the blank condition field and select the Goal for Left Paddle actor from the list.

ON Rule: Left Paddle Scored

If of the following conditions are valid

this actor receives event actor of type

🔍 Type down key to select condition or start typing to search.

Then

Drop behaviors here.

Else

22. Locate the change attribute behavior and drag that into the Left Paddle Scored rule.
23. Rename the Change Attribute Behavior to 'Change Attribute: Increase LeftScore'

ON Change Attribute: Increase LeftScore

Change attribute to ()

Errors

Please choose an attribute to change.

24. Click in the first blank field for the change attribute behavior and select game.LeftScore as the attribute to change.

ON

Change Attribute: Increase LeftScore

Change attribute

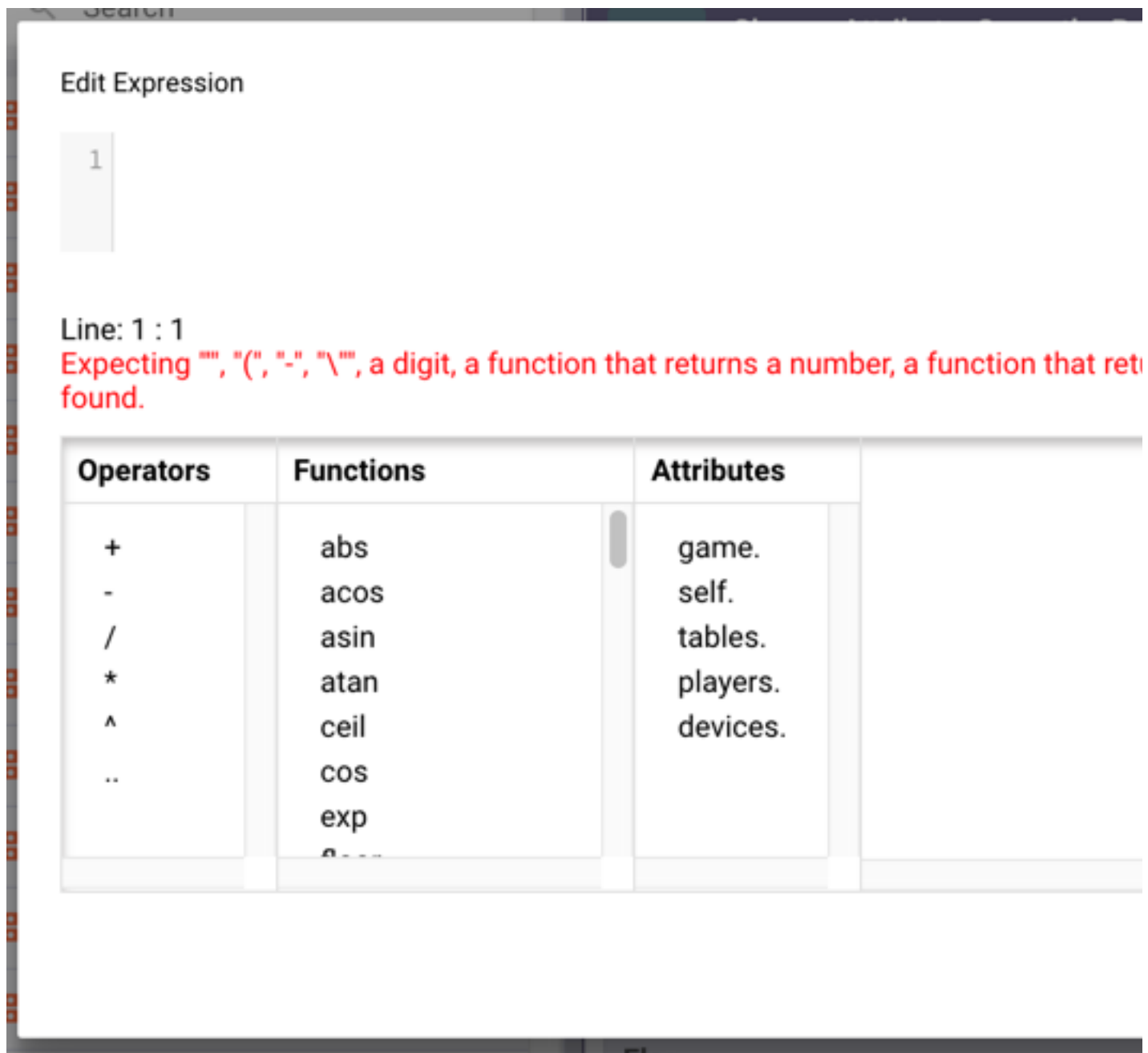
game.LeftScore

 to ()

Errors

The attribute expects a integer value.

25. Click the open parenthesis '(' to convert the next field to an expression, then click on the empty field to open the expression editor.



26. Add the expression 'game.LeftScore + 1' to the expression editor and click the update button to save it.

Search

Edit Expression

1

game.LeftScore+1

Line: 1 : 17


| Operators | |
|---|---|
| <div><div>+</div><div>-</div><div>/</div><div>*</div><div>^</div></div> | <div>abs <i>abs(number)</i></div> <div>Returns the "absolute value" of a number. In other words, it will For example, $\text{abs}(-5.23) = 5.23$.</div> |

27. Drag a destroy behavior into the Left Paddle Scored rule, beneath the change attribute behavior.

The screenshot displays a logic editor interface. At the top, a dark purple header bar contains a green 'ON' button and the title 'Rule: Left Paddle Scored'. Below this, a light purple bar states 'If all of the following conditions are valid'. The first condition is 'this actor receives event overlaps or collides actor of type Goal for Left Paddle'. A search bar with a magnifying glass icon and the text 'Type down key to select condition or start typing to search.' is positioned below the conditions. The 'Then' section follows, containing two stacked rule blocks. The first block, titled 'Change Attribute: Increase LeftScore', has a green 'ON' button and the text 'Change attribute game.LeftScore to f(x) { game.LeftScore+1 }'. The second block, titled 'Destroy', has a green 'ON' button and the text 'Destroy this actor'. Both rule blocks have a blue glow effect at their bottom edges.

Now when the ball collides with the Goal for Left Paddle actor, the ball will increase the LeftScore attribute by 1 and destroy itself. Let's set up the same kind of rule so that the right paddle can score too.

28. Add a rule to the ball actor by clicking the blue add rule button across the top of the logic stack.
29. Rename the rule to 'Rule: Right Paddle Scored'.

ON Rule: Right Paddle Scored 

If **all** of the following conditions are valid

this actor receives event **mouse pointer** **over actor**

🔍 Type down key to select condition or start typing to search.

Then

Drop behaviors here.

Else

30. Change the condition of the rule from 'mouse pointer' to 'overlaps or collides'.

31. Click in the blank condition field and select the Goal for Right Paddle actor from the list.

ON Rule: Right Paddle Scored


If **all** of the following conditions are valid

this actor receives event **overlaps or collides** **actor of type** **Goal for Right Paddle**

🔍 Type down key to select condition or start typing to search.

32. Locate the change attribute behavior and drag that into the Right Paddle Scored rule.

33. Rename the Change Attribute Behavior to 'Change Attribute: Increase RightScore'

ON Change Attribute: Increase RightScore 

Change attribute to ()

Errors

Please choose an attribute to change.

34. Click in the first blank field for the change attribute behavior and select game.RightScore as the attribute to change.

ON Change Attribute: Increase RightScore

Change attribute to ()

Errors

The attribute expects a integer value.

35. Click the open parenthesis '(' to convert the next field to an expression, then click on the empty field to open the expression editor.
36. Add the expression 'game.RightScore + 1' to the expression editor and click the update button to save it.

```
1 game.RightScore+1
```

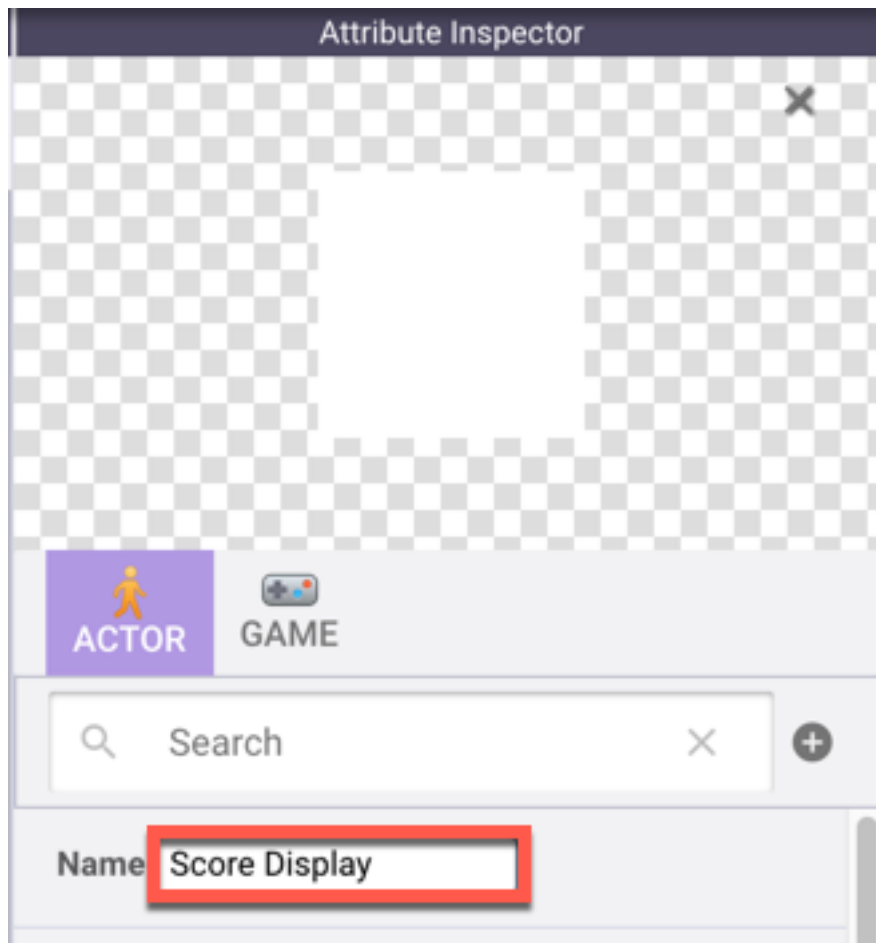
| Operators | |
|---|--|
| <div><div>+</div><div>-</div><div>/</div><div>*</div><div>^</div></div> | <div>acos</div> <div><i>acos(number)</i></div> <div>This is the trigonometric arccosine (inverse cosine) function. Values outside the range [-1, 1] will result in 'nan'.</div> |

37. Drag a destroy behavior into the Right Paddle Scored rule, beneath the change attribute behavior.

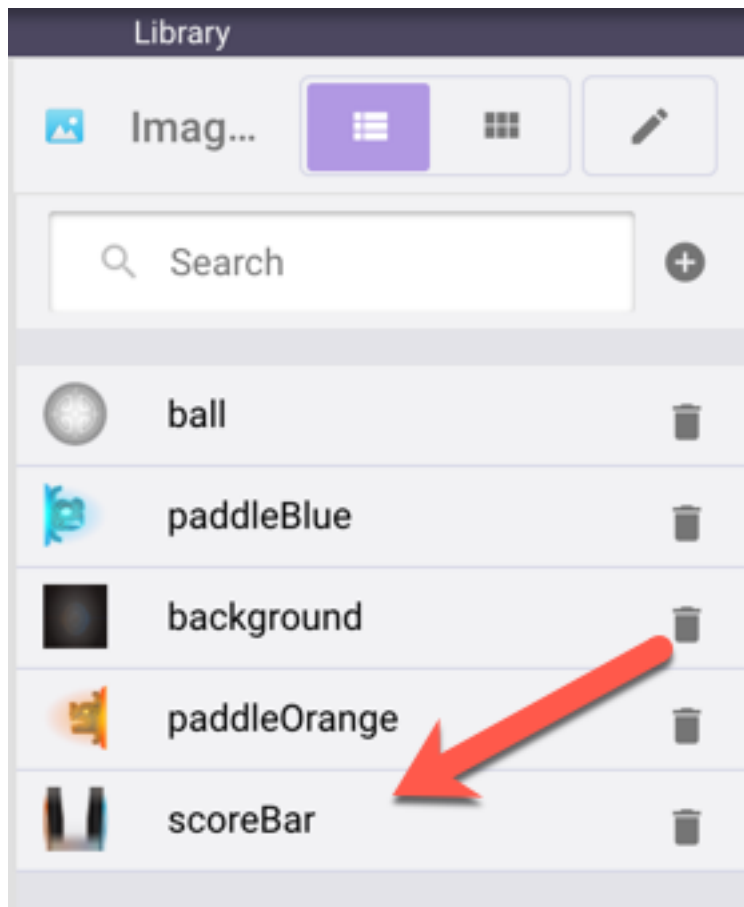


Now that we have our logic all set up, we should test it. But in order to test it we'll need to have some form of display that shows both of our score attributes.

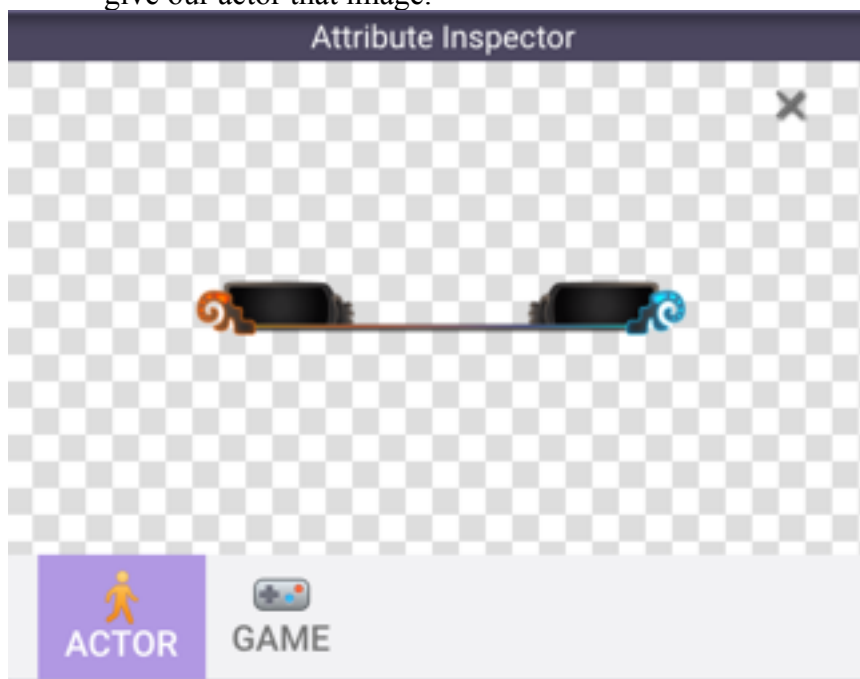
38. Click on the Actors tab in the Library and click the '+' button next to the search bar to create a new Actor.
39. Click on the newly created actor to open the Actor Editor.
40. Locate the 'name' attribute for the actor in the inspector over on the right side of the screen, and rename the actor to 'Score Display'.



41. Click on the images tab in the library and locate the 'scoreBar' image.

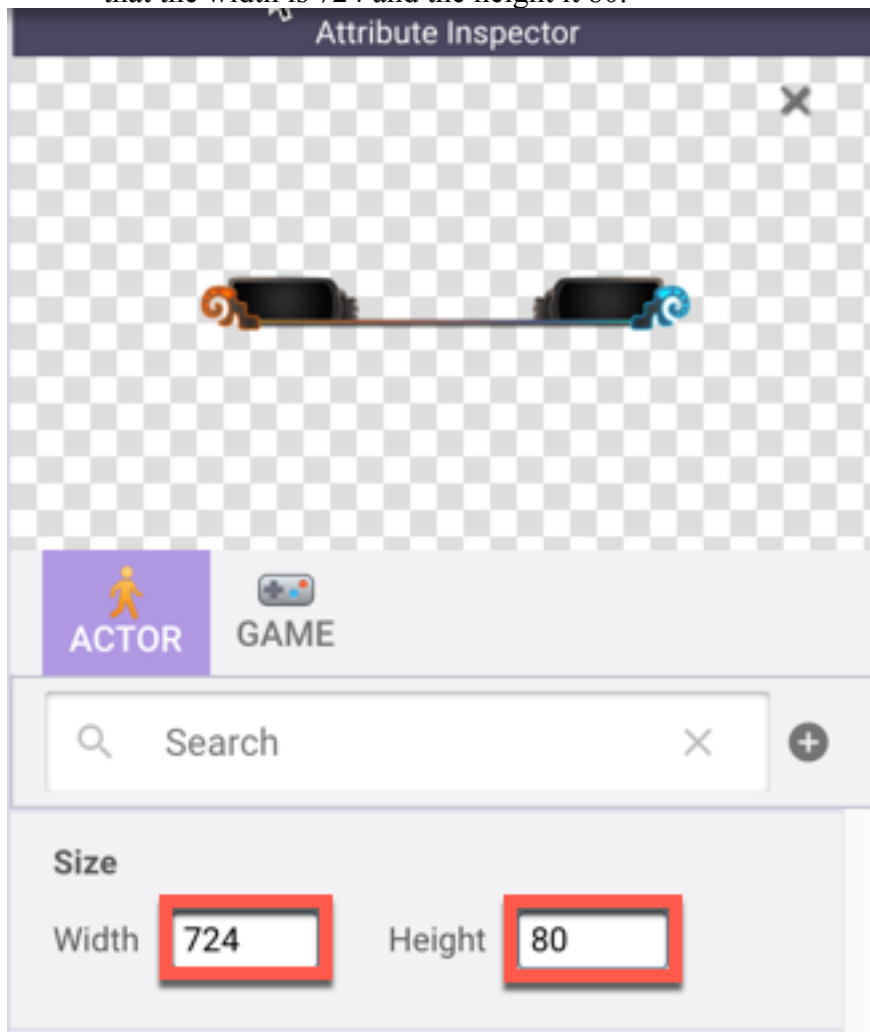


42. Drag the 'scoreBar' image into the actor image area in the top part of the inspector to give our actor that image.

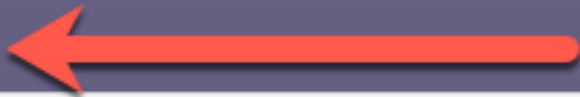


43. In the inspector on the right (make sure you have the actor tab selected), double check




that the width is 724 and the height it 80.



44. Click on the behaviors tab in the library and locate the display text behavior.
45. Drag the display text behavior into the logic stack and rename it 'Display Text: Left Paddle Score'.


ON **Display Text: Left Paddle Score** 

Text:
" Hello world! "

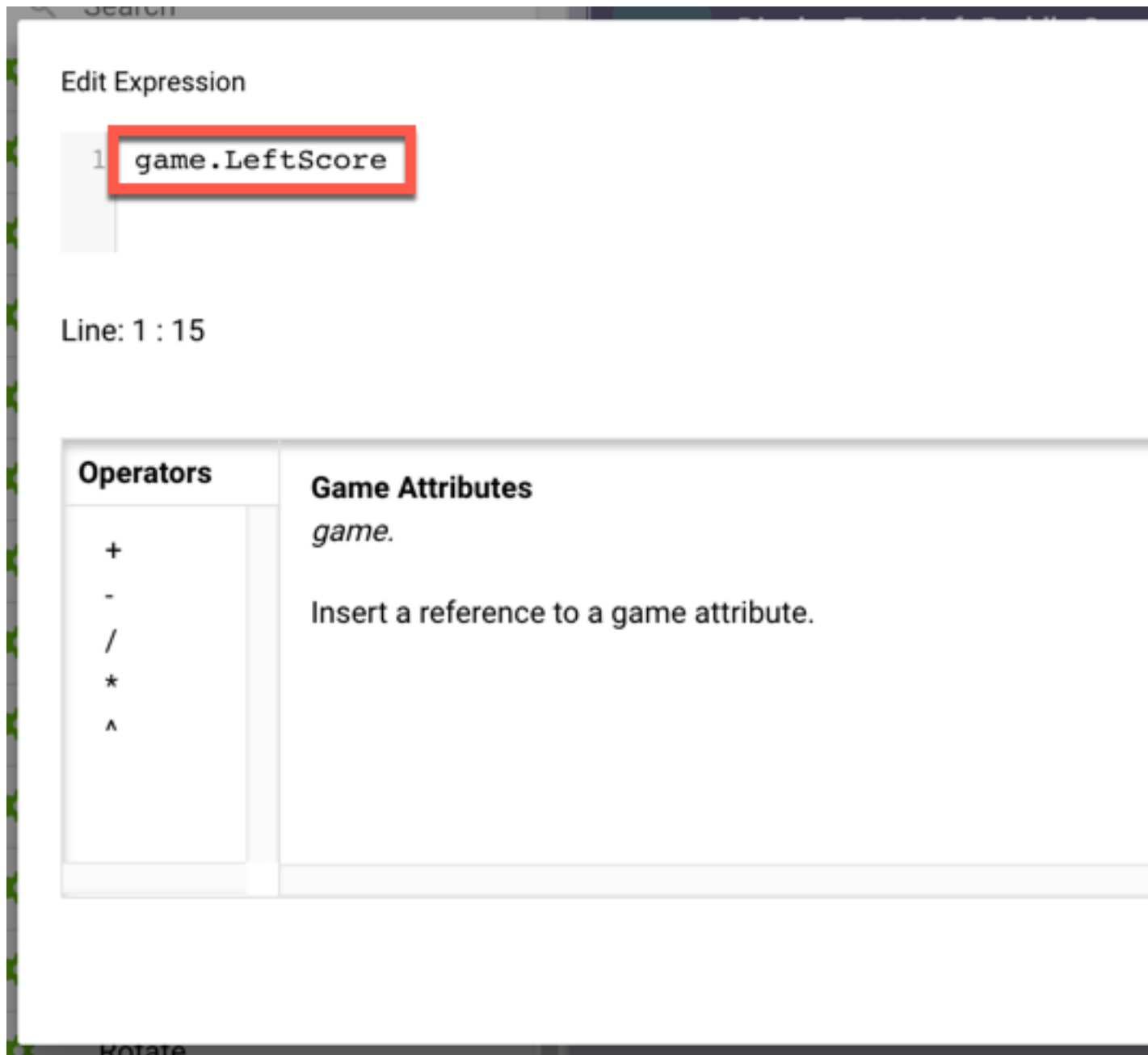
   ☐ Wrap inside actor

place at ↔ (0) ↑↓ (0) relative to actor ▼

at an angle of (0) ↻ relative to actor ▼

using font Cambay (* Arial) ▼ with size 30 and color 

46. Click the quotes (‘’) button to change the text being displayed to an expression and click the blank expression field to open the expression editor.
47. Add the attribute ‘game.LeftScore’ to the expression and click update to save it.



48. Click on the scene tab in the library and select the Gameplay scene to open the scene editor.
49. Click on the actors tab in the library and drag an instance of the Score Display actor onto the scene and position it in the middle of the screen, towards the top.






Preview the game to see how the score display looks with the default values. You should notice that the text isn't quite where we want it, and with the color being set to black, it's very hard to see the value.

50. Click on the actors tab in the library and select the Score Display actor.
51. Inside the display text behavior, for the first "place at" field (which represents the x position of the text), fill in '-240'.

ON Display Text: Left Paddle Score

Text:
`f(x) {` `}`

   ☐ Wrap inside actor

place at ↔ () ↑↓ () relative to ▼




at an angle of () ↻ relative to ▼

using font ▼ with size and color ☐

52. Change the size of the text to 45 so that it's a little bigger.

ON Display Text: Left Paddle Score

Text:
`f(x) {` `}`

   ☐ Wrap inside actor

place at ↔ () ↑↓ () relative to ▼

at an angle of () ↻ relative to ▼

using font ▼ with size and color ☐

53. Click on the black square to change the color of the text to orange (since this is the text representing our left paddle which is also orange).

ON Display Text: Left Paddle Score

Text:
 f(x) { `game.LeftScore` }

☐ ☒ ☐ Wrap inside actor

place at ↔ () ↕ () relative to

at an angle of () ↻ relative to

using font with size and color

54. Copy and paste the display text behavior, and rename it “Display Text: Right Paddle Score”.

ON Display Text: Right Paddle Score ←

Text:
 f(x) { `game.LeftScore` }

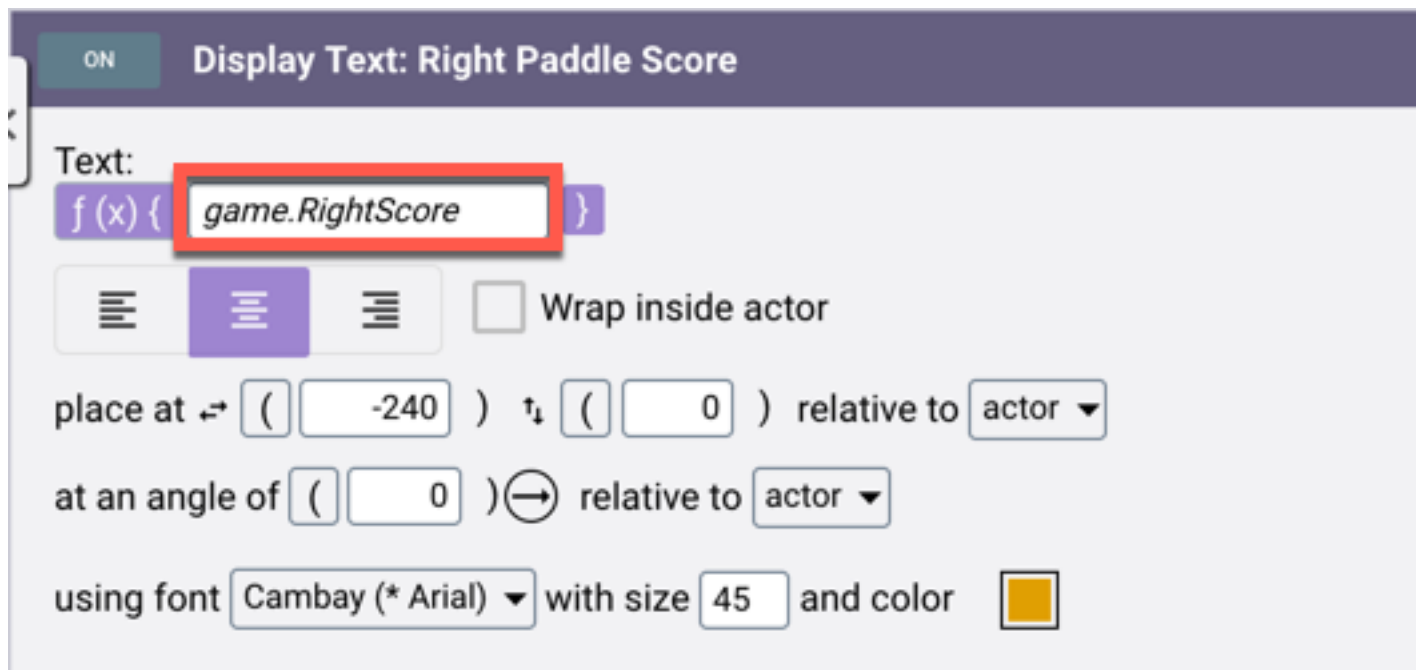
☐ ☒ ☐ Wrap inside actor

place at ↔ () ↕ () relative to

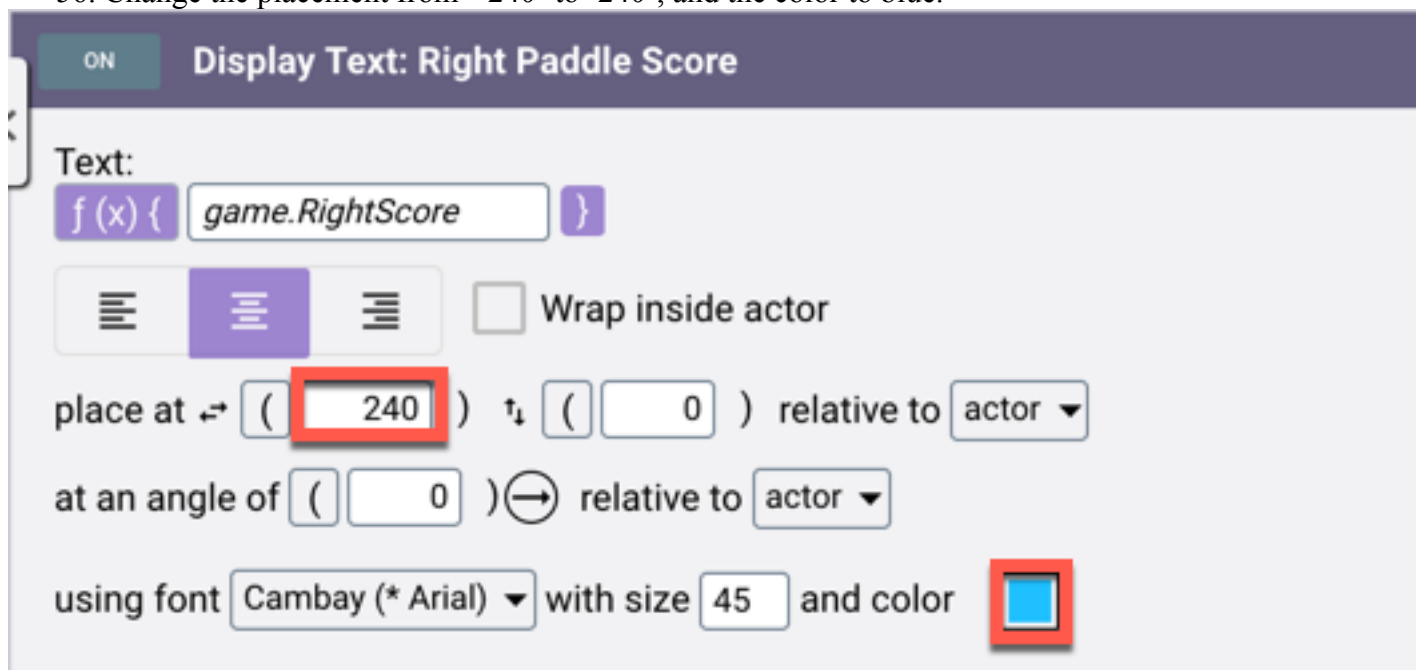
at an angle of () ↻ relative to

using font with size and color

55. Click in the expression field and change the attribute that we’re displaying to “game.RightScore”.



56. Change the placement from '-240' to '240', and the color to blue.

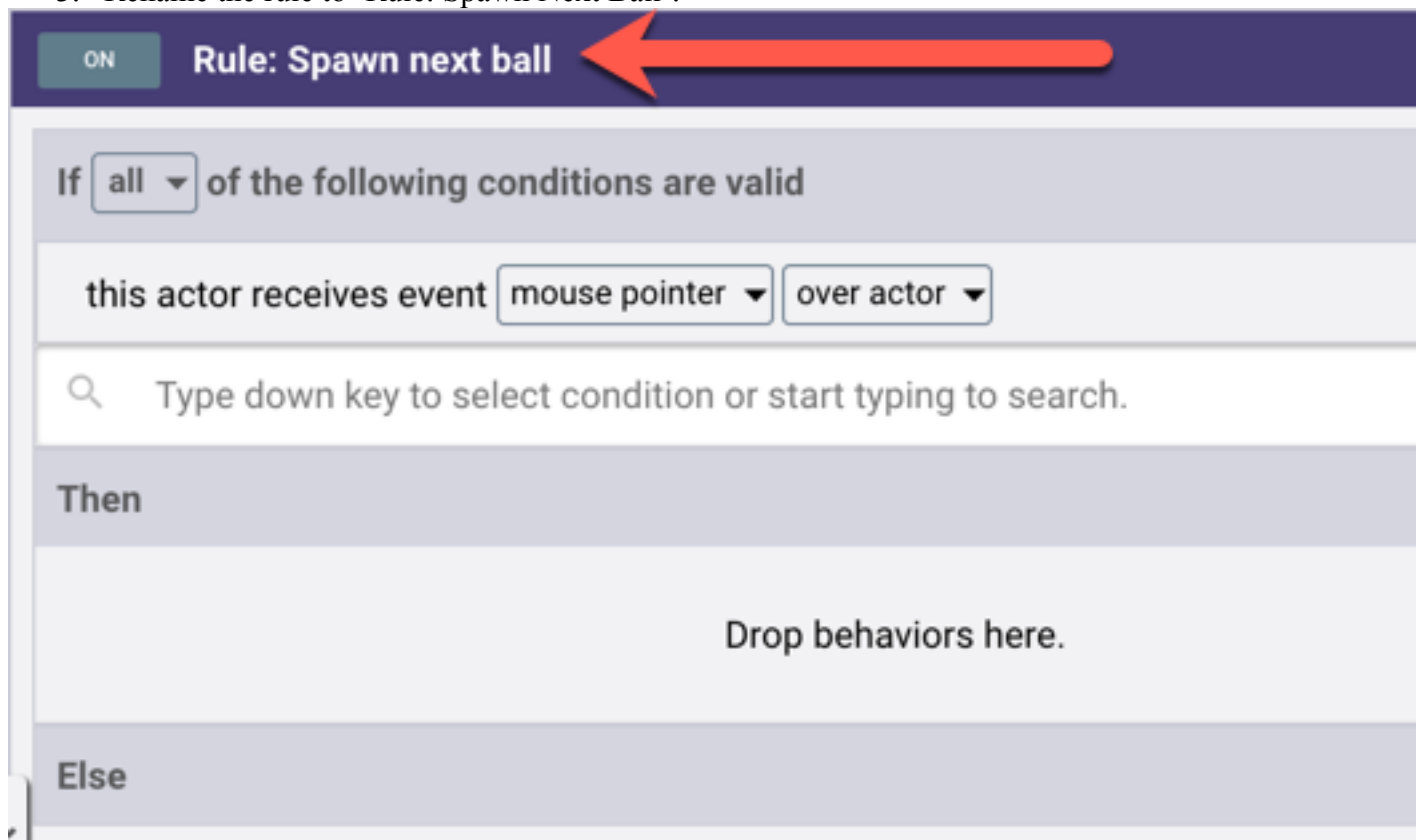


Preview the game and let each paddle score to make sure that all the logic is working correctly and that the scores are being displayed nicely.

Spawning More Balls

You may have noticed that once one of the paddle's has scored, no new ball appears. Let's fix that!

1. Click on the actors tab in the library and select the Goal for Left Paddle actor.
2. Add a rule to the actor by clicking the blue add rule button across the top of the logic stack.
3. Rename the rule to 'Rule: Spawn Next Ball'.



4. Change the condition of the rule from 'mouse pointer' to 'overlaps or collides'.
5. Click in the blank condition field and select the Ball actor from the list.



6. Click on the behaviors tab in the library and locate the timer behavior.

7. Drag a timer behavior into the then section of the rule.

ON Rule: Spawn next ball

If **all** of the following conditions are valid

this actor receives event **overlaps or collides** actor of type **Ball**

Type down key to select condition or start typing to search.

Then

ON Timer

Every (5) seconds... ☐ Run to completion.

Do

Drop behaviors here.

8. Change the timer from 'every' to 'after', replace the seconds value with '2', and check the run to completion box.

ON Timer


After (2) seconds... ☒ Run to completion.

Do

Drop behaviors here.

9. In the behaviors tab of the library, locate the spawn actor behavior and drag it into the timer.

10. Click the blank field of the spawn actor behavior and select the ball actor.



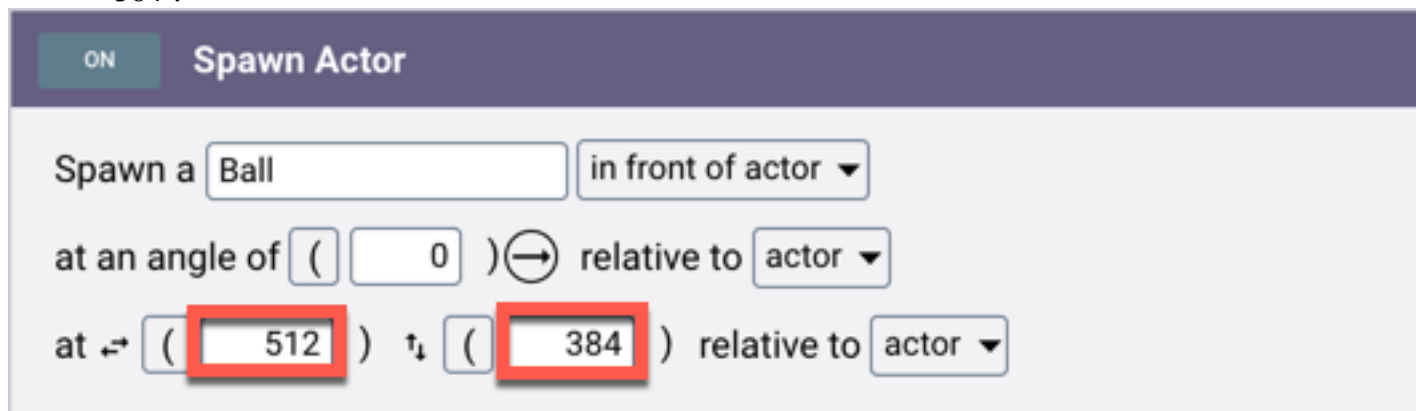
ON Spawn Actor

Spawn a **Ball** in front of actor ▼

at an angle of (0) (→) relative to actor ▼

at ↔ (0) ↕ (0) relative to actor ▼

11. Set the x (left/right) value of the spawn actor behavior to '512', the y (up/down) value to '384'.



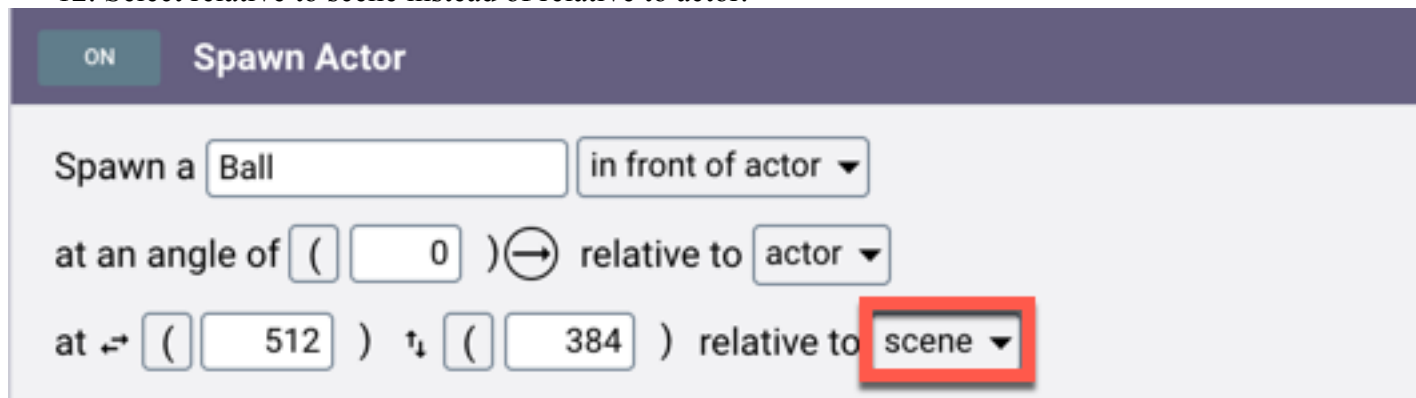
ON Spawn Actor

Spawn a **Ball** in front of actor ▼

at an angle of (0) (→) relative to actor ▼

at ↔ (512) ↕ (384) relative to actor ▼

12. Select relative to scene instead of relative to actor.



ON Spawn Actor

Spawn a **Ball** in front of actor ▼

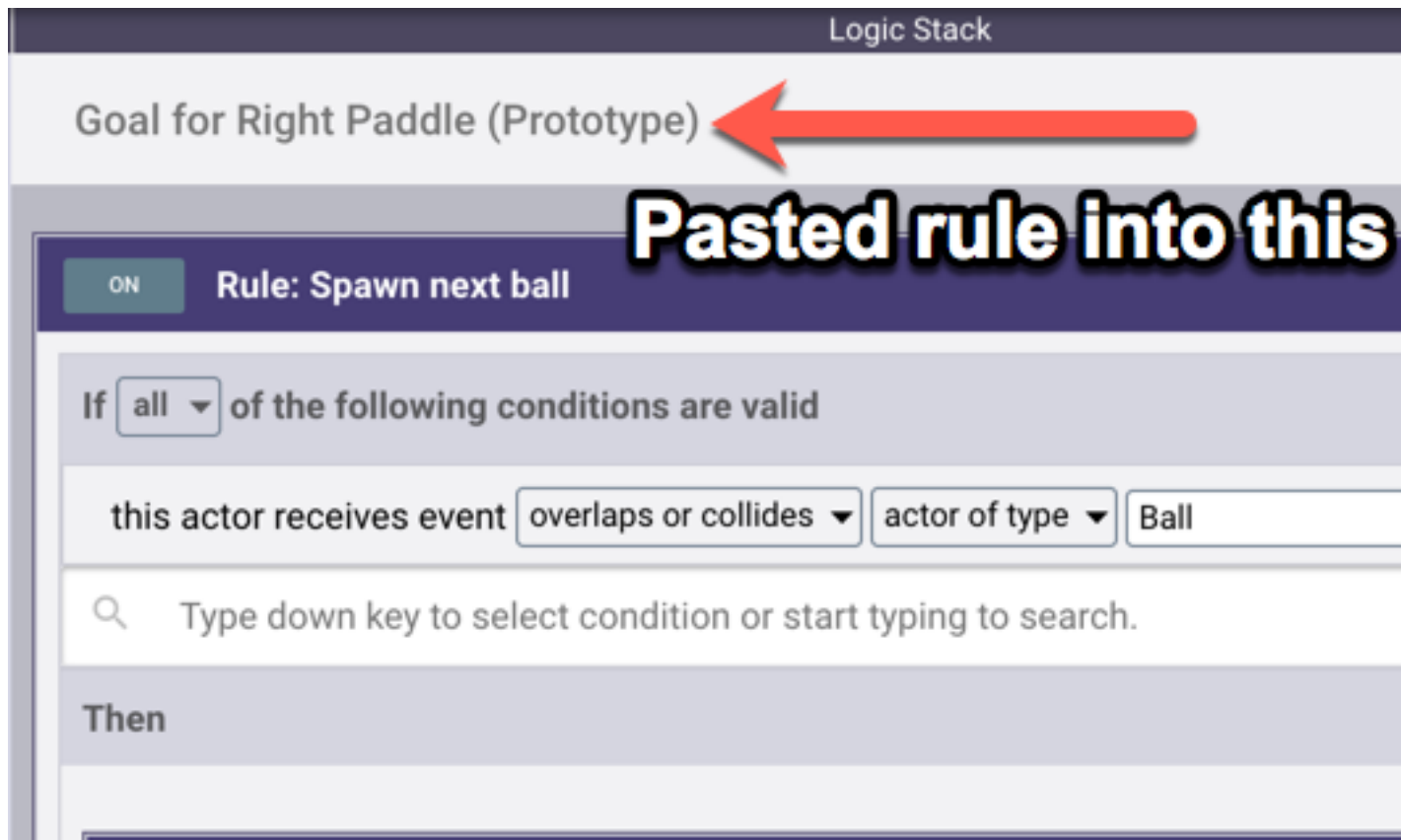
at an angle of (0) (→) relative to actor ▼

at ↔ (512) ↕ (384) relative to **scene** ▼

13. Select the rule and use the keyboard shortcut to copy it (control+c or command+c).

14. Click on the actors tab in the library and select the goal for right paddle actor.

15. Use the keyboard shortcut to paste the Span Next Ball rule into the actor (control+v or command+v).



Preview the game and let both paddles score to make sure that a new ball is correctly spawning.

Adding AI for the Right Paddle

It's not very fun having just one player control both paddles, so let's make some logic to have the right paddle played by the computer.

First, we'll need to keep track of the balls x velocity (the direction it's moving) and y position so that the right paddle knows where it needs to move.

1. Click on the actors tab in the library and select the ball actor.
2. In the inspector (on the right side of the screen) click the game tab to view and create game attributes.
3. Click the + button next to the search bar to create two new 'real' game attributes.
4. Rename them 'BallXVelocity' and 'BallYPosition'.

| | | | |
|---------------|---|---------|---|
| RightScore | 0 | integer | — |
| BallXVelocity | 0 | real | — |
| BallYPosition | 0 | real | — |

5. In the behaviors tab of the library, locate the constrain attribute behavior. Drag it into the logic stack (make sure it's outside of any rules).
6. Rename the behavior to “Constrain Attribute: Track BallXVelocity”.

ON
Constrain Attribute: Track BallXVelocity

Constrain attribute
Select attribute
to value (of) (
select an attribute to c
)

Errors

Please choose a target attribute.

7. For the first attribute field, fill in game.BallXVelocity.

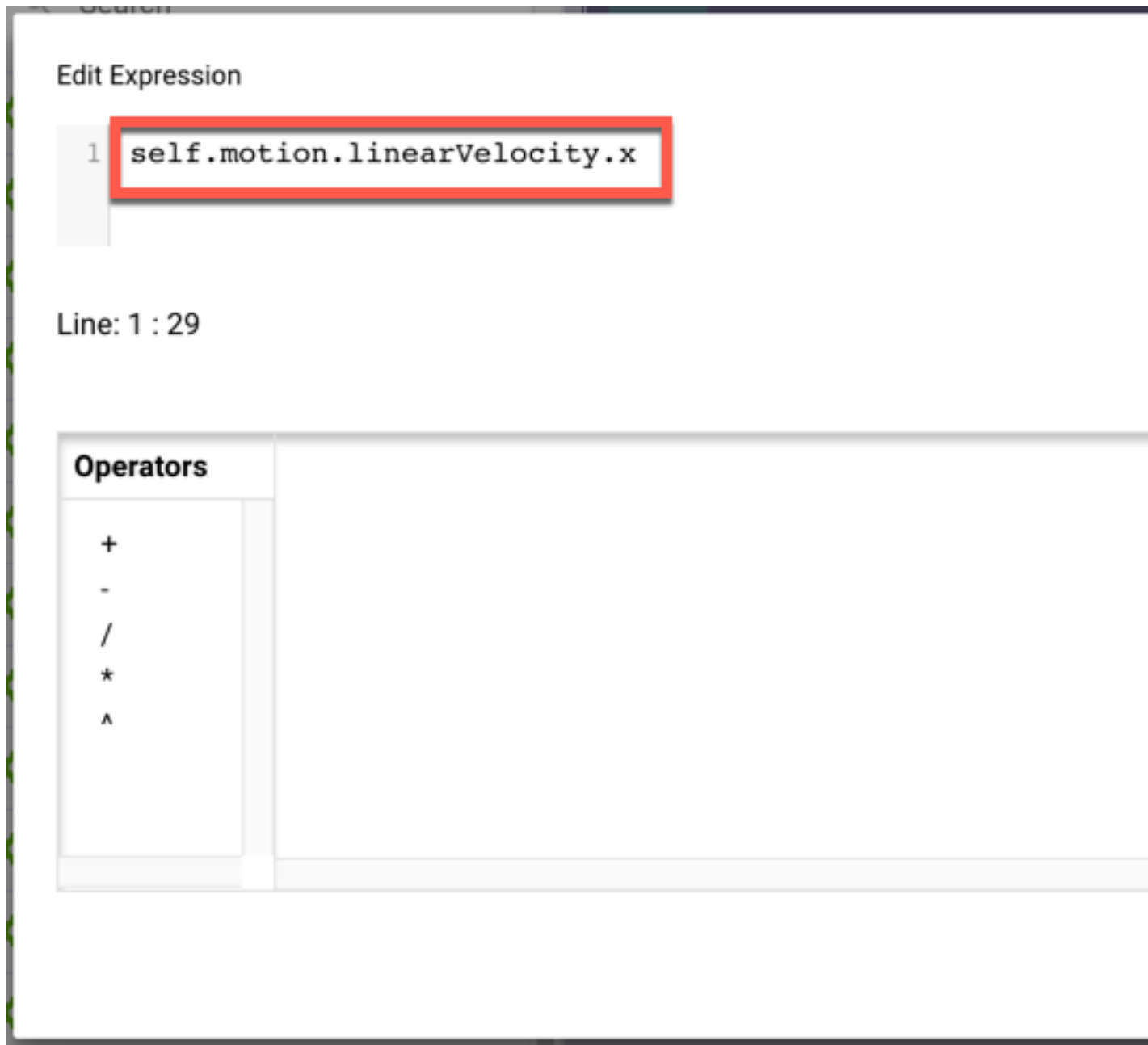
ON
Constrain Attribute: Track BallXVelocity

Constrain attribute
game.BallXVelocity
to value (of) (
)

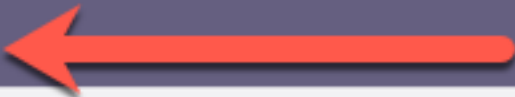
Errors

Constraint value is of wrong type. A value of type real is expected.

8. Click on the open parenthesis to convert the next field to an expression, and click on the empty field to open the expression editor.
9. In the expression editor, fill in self.motion.linearVelocity.x and click update to save the expression.



10. Drag another constrain attribute behavior into the logic stack and rename it "Constrain Attribute: Track BallYPosition".

ON **Constrain Attribute: Track BallYPosition** 

Constrain attribute to value (of) ()

Errors

Please choose a target attribute.

11. For the first attribute field, fill in game.BallYPosition.

ON **Constrain Attribute: Track BallYPosition**

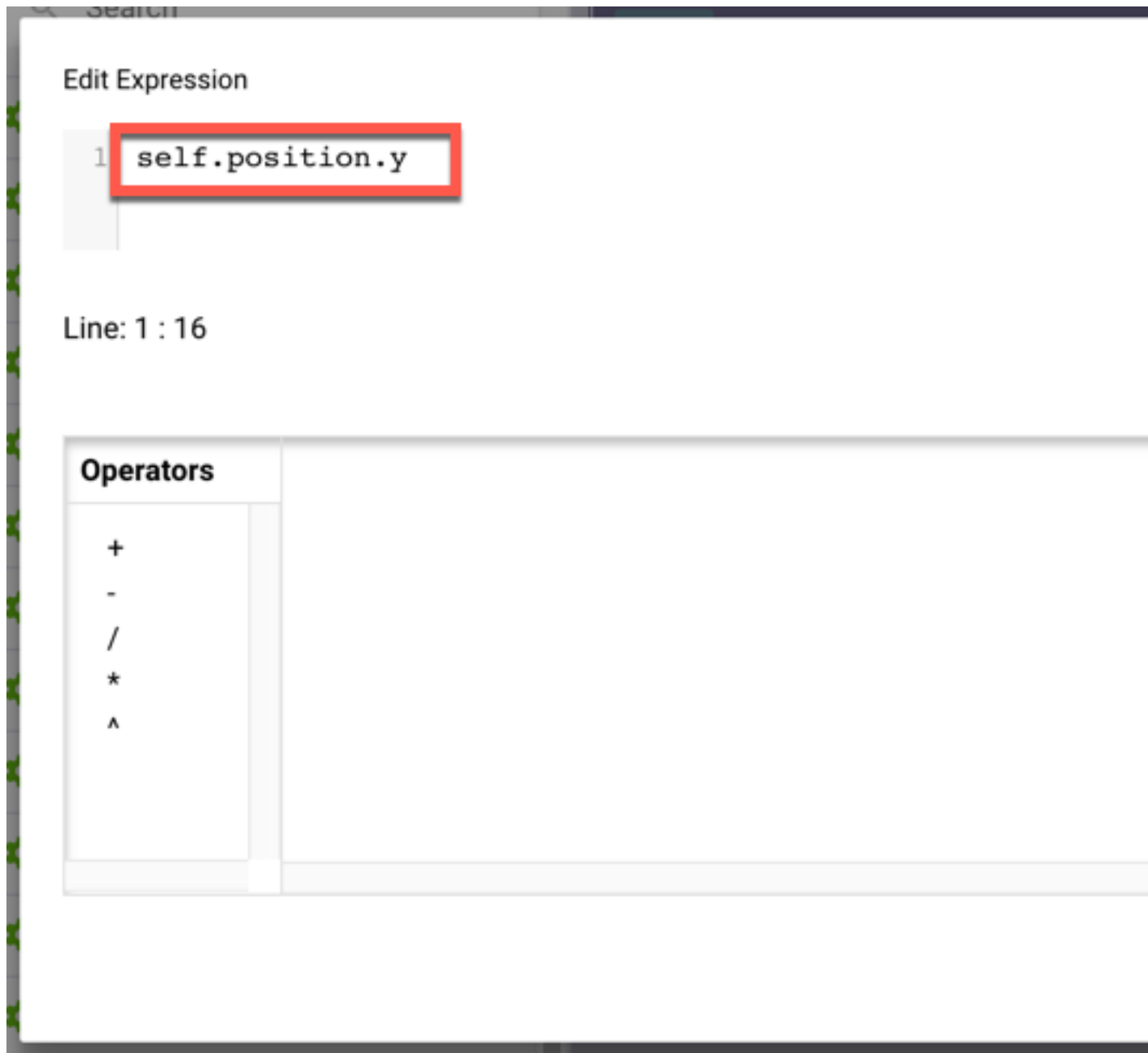
Constrain attribute to value (of) ()

Errors

Constraint value is of wrong type. A value of type real is expected.

12. Click on the open parenthesis to convert the next field to an expression, and click on the empty field to open the expression editor.

13. In the expression editor, fill in self.position.y and click update to save the expression.




Next we'll need to rework the motion of the right paddle so that it's not player controlled.

14. Click on the actors tab in the library and select the Right Paddle actor.
15. Inside the Move Down rule, delete the keyboard key condition.
16. Click in the blank condition selection field and select attribute comparison to add an attribute condition.

ON Rule: Move Down

If **all** of the following conditions are valid

the attribute  **New condition**

17. Click inside the blank attribute field and select game.BallYPosition.

ON Rule: Move Down

If **all** of the following conditions are valid

the attribute =

18. Change the '=' to '<' and click the open parenthesis to change the next field to an expression.

19. Click on the blank expression field to open the expression editor.

20. Fill in self.position.y-64 and click the update button to save the expression.

ON Rule: Move Down

If **all** of the following conditions are valid

the attribute <

64 is half of the paddles height, so self.position.y-64 is the y position of the bottom of the paddle.

21. Add another attribute comparison condition to the rule.

ON Rule: Move Down

If all of the following conditions are valid

the attribute < f (x) { }

the attribute

🔍 Type down key to select condition or start typing to search.

22. Click inside the blank attribute field of the rule and select game.BallXVelocity.
23. Change the '=' to '>' and fill in the new field with a '0' (this means that the ball is moving to the right).

ON Rule: Move Down

If all of the following conditions are valid

the attribute < f (x) { }

the attribute > ()

🔍 Type down key to select condition or start typing to search.

24. Change the speed value of the move behavior inside the rule from 300 to 500.

ON Move

Move in direction () ↓ relative to actor at speed ()

Movement is additive

We're going to do similar steps with the Move Up rule as well.

25. Delete the keyboard key condition.
26. Click in the blank condition selection field and select attribute comparison to add an attribute condition.
27. Click inside the blank attribute field and select game.BallYPosition.
28. Change the '=' to '>' and click the open parenthesis to change the next field to an expression.
29. Click on the blank expression field to open the expression editor.
30. Fill in self.position.y+64 and click the update button to save the expression.

ON Rule: Move Up

If of the following conditions are valid

the attribute

🔍 Type down key to select condition or start typing to search.

64 is half of the paddles height, so self.position.y+64 is the y position of the top of the paddle.

31. Add another attribute comparison condition to the rule.
32. Click inside the blank attribute field of the rule and select game.BallXVelocity.
33. Change the '=' to '>' and fill in the new field with a '0' (this means that the ball is moving to the right).

ON Rule: Move Up

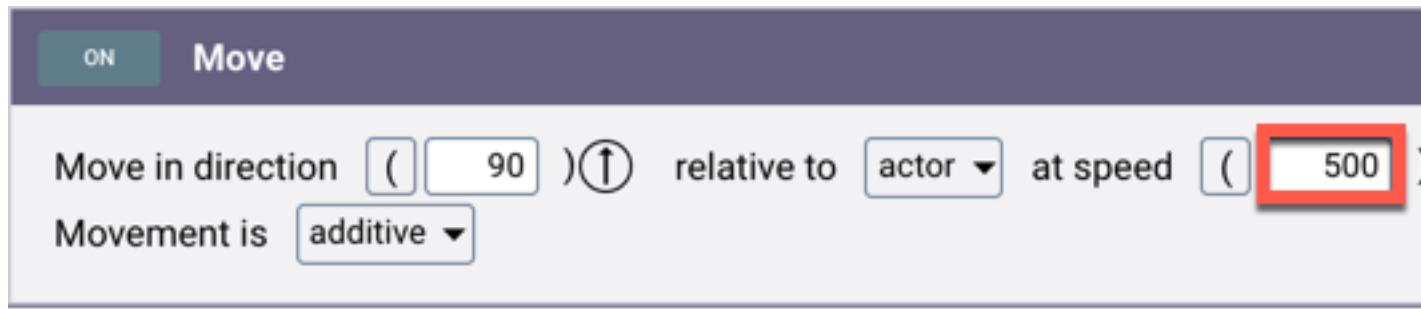
If of the following conditions are valid

the attribute

the attribute

🔍 Type down key to select condition or start typing to search.

34. Change the speed value of the move behavior inside the rule from 300 to 500.



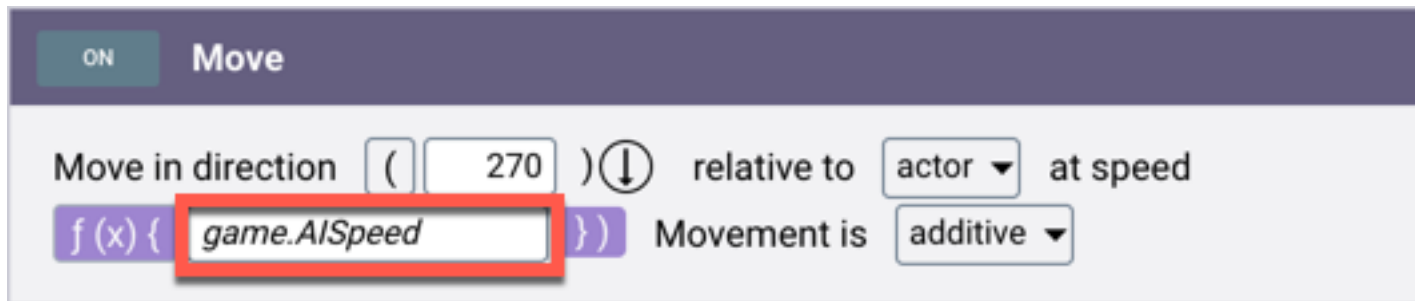
Preview the game and make sure the right paddle automatically moves up/down to catch the ball. If the paddle isn't moving, that's because the paddle has no reason to move because the ball is already on a path to collide with the paddle. Try moving the ball higher up the scene where the AI paddle will need to move to hit it. You may notice that the paddle is moving a little fast, which makes it hard to score against the computer.

What we can do to fix that is have the speed of the paddle set through an attribute, and decrease the speed over time.

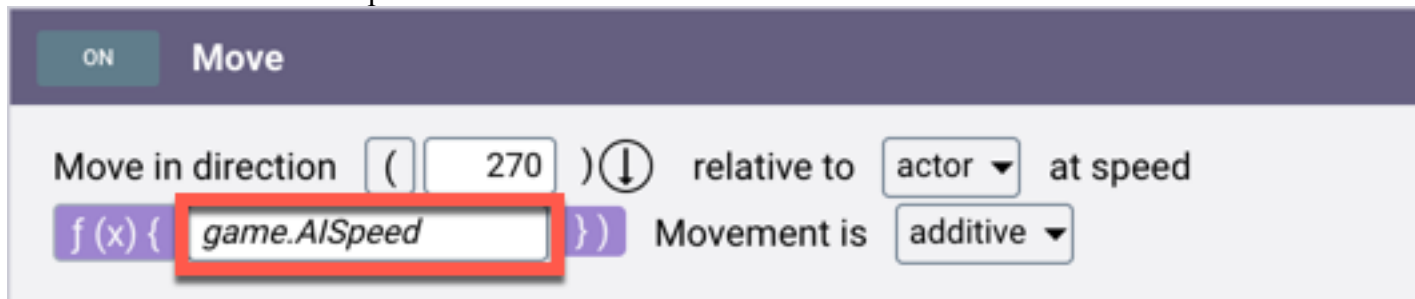
35. Inside the Actor Editor for the RightPaddle, locate the inspector section on the right side of the screen.
36. Click on the game tab in the inspector, and click the + button to create a new 'real' game attribute.
37. Rename the attribute to AISpeed and set the value to 500.



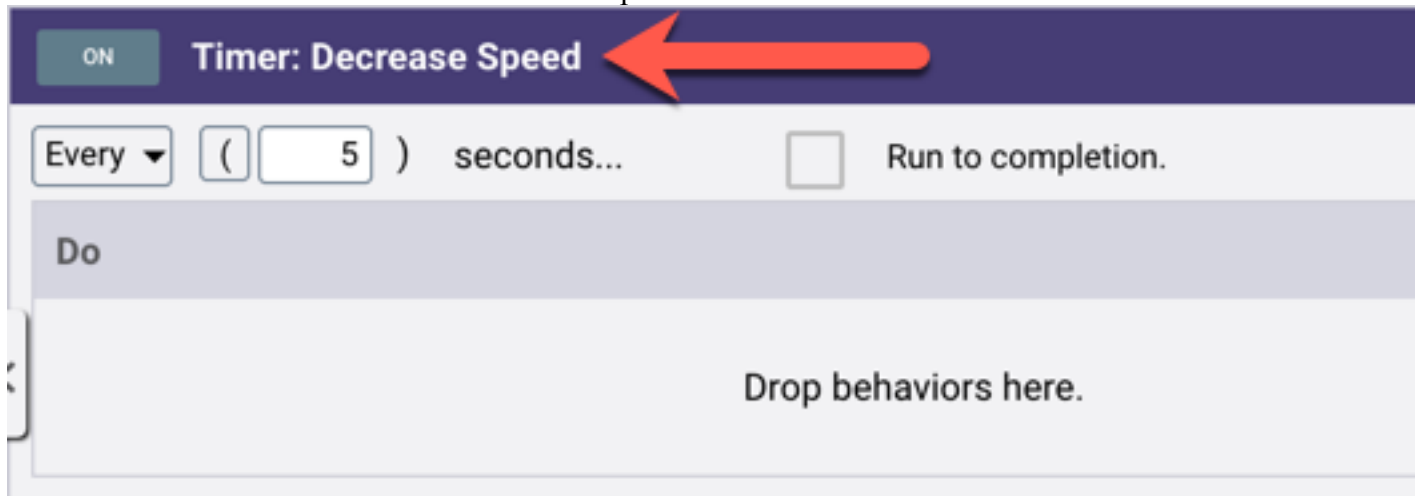
38. Inside the Move behavior that's in the Move Down rule, change the speed value to an expression (by clicking the open parenthesis).
39. Open the expression editor for the speed value, fill in game.AISpeed, and click the update button to save the expression.



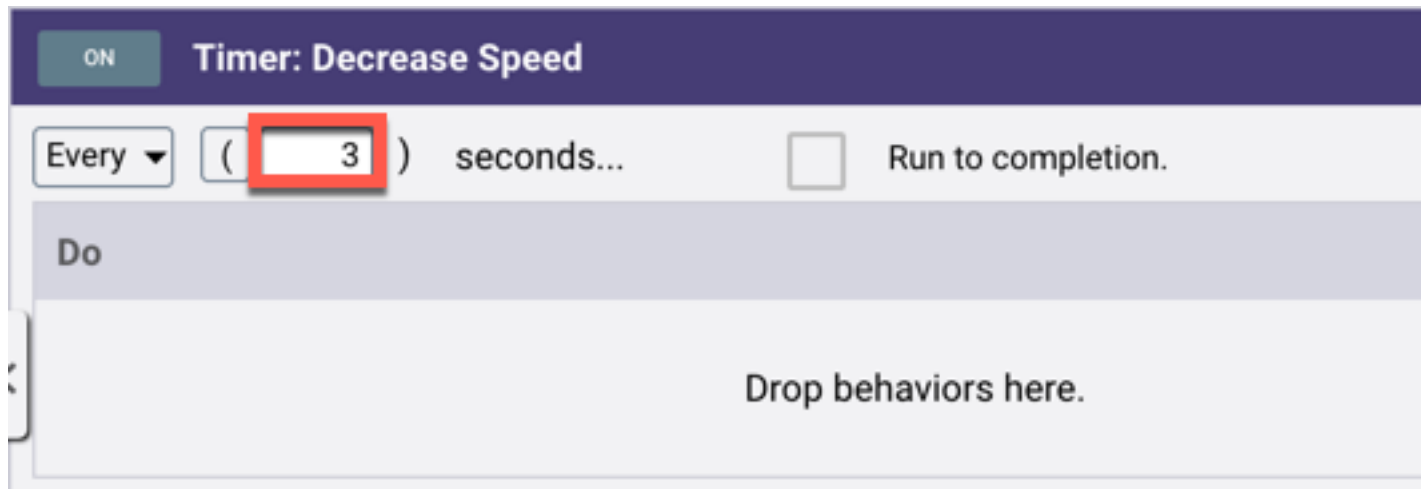
40. Inside the Move behavior that's in the Move Up rule, change the speed value to an expression (by clicking the open parenthesis).
41. Open the expression editor for the speed value, fill in `game.AISpeed`, and click the update button to save the expression.



42. Add a timer to the logic stack by clicking the timer button in the logic stack bar.
43. Rename the timer to 'Timer: Decrease Speed'.

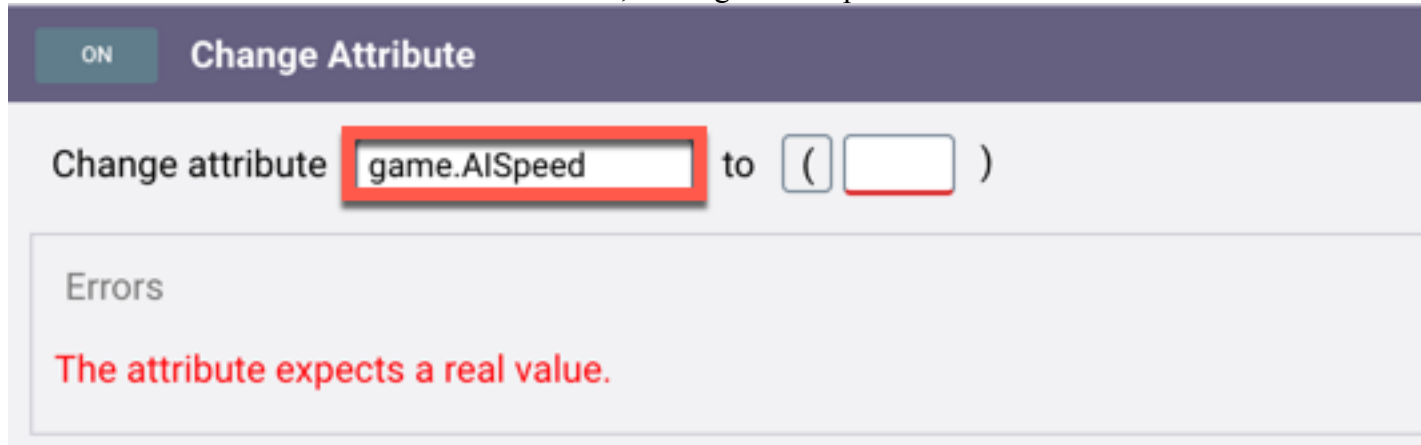


44. Leave the type of timer set to 'every' and change the seconds value to '3'.



45. Select the behaviors tab in the library and locate the change attribute behavior. Drag one into the timer.

46. In the first attribute field of the behavior, fill in game.AISpeed.



47. Convert the second field to an expression, and open the expression editor.

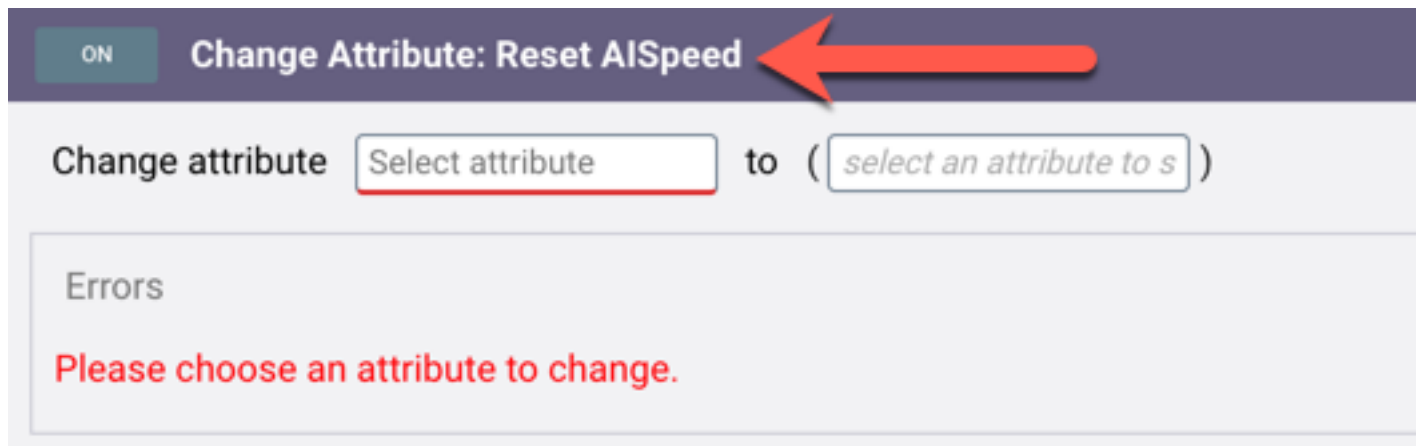
48. Type the expression 'game.AISpeed - 20' and click the update button to save it.



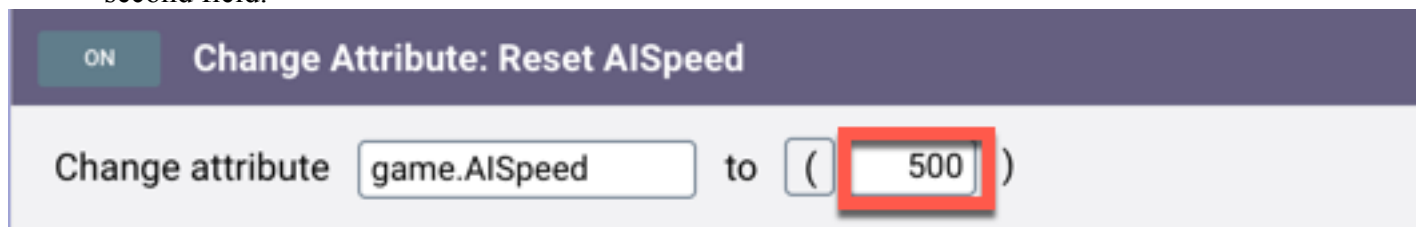
This will ensure that the computer controlled paddle slows down by 20 every 3 seconds. Preview the game and make sure that the right paddle slows down over time. You may have noticed that the paddle never speeds back up. Let's fix that by resetting the AISpeed attribute when a goal is made.

49. Open the actor editor for the Goal for Left Paddle actor.

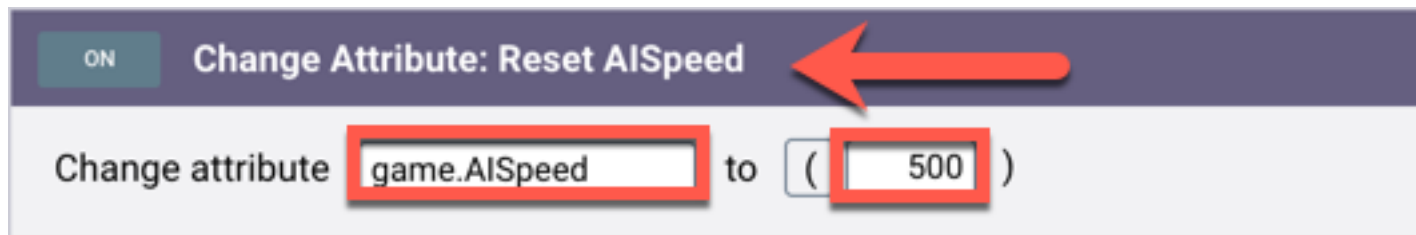
50. Add a change attribute behavior beneath the spawn actor behavior inside the timer and rename it 'Change Attribute: Reset AISpeed'.



51. Select game.AISpeed for the first attribute field in the behavior, and fill in '500' for the second field.



52. Open the actor editor for the Goal for Right Paddle actor.
53. Add a change attribute behavior beneath the spawn actor behavior inside the timer and rename it 'Change Attribute: Reset AISpeed'.
54. Select game.AISpeed for the first attribute field in the behavior, and fill in '500' for the second field.



Preview the game again and wait for the paddle to slow down, then score a point to make sure the speed resets.

Create Random Serving for the Ball

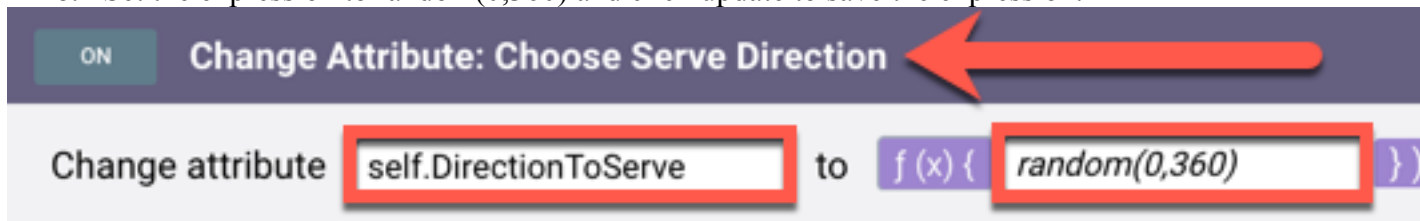
Right now the ball is always served directly to the right, which isn't very exciting. Let's change it so that the ball is spawned in a random direction.

1. Open the actor editor for the Ball actor.
2. In the inspector on the right side of the screen (make sure the actor tab is selected), click the + button next to the search bar to create a 'real' actor attribute.

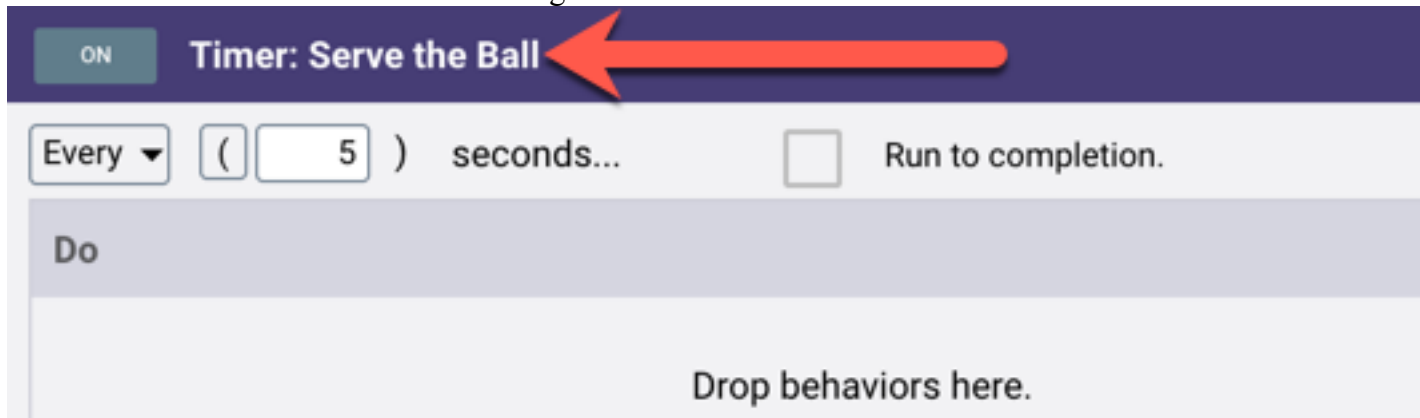
3. Rename the attribute “DirectionToServe”.



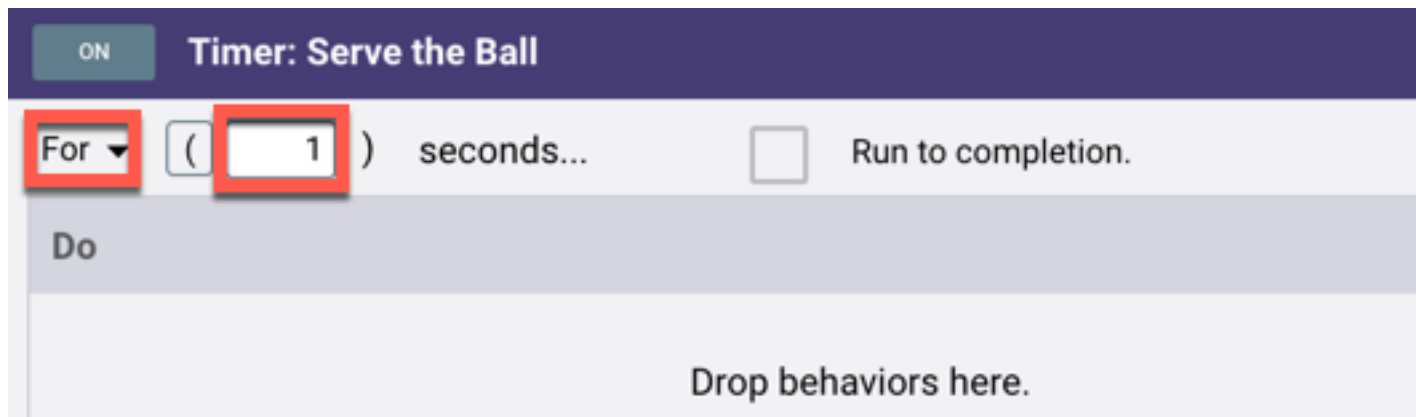
4. Delete the change attribute behavior currently being used to serve the ball.
5. Add a new change attribute behavior to the top of the logic stack and rename it to “Change Attribute: Choose Serve Direction”
6. For the first attribute field in the behavior select self.DirectionToServe.
7. Convert the second field to an expression and open the expression editor.
8. Set the expression to random(0,360) and click update to save the expression.



9. Add a timer to the bottom of the logic stack and rename it “Timer: Serve the Ball”.



10. Change the type of timer from “every” to “for” and set the seconds value to 1.




11. Add an accelerate behavior inside the newly created timer.
12. Change the direction field of the accelerate behavior into an expression and open the expression editor.
13. Add the attribute “self.DirectionToServe” to the expression and click the update button to save it.
14. Set the rate value of the accelerate behavior to 700.





Preview the game to make sure the ball is being served correctly in random directions. The ball is likely going a little too fast, but we can fix this through the use of some motion attributes.


15. Inside the actor editor for the ball actor, locate and expand the list of motion attributes at the bottom of the Inspector.


Attribute Inspector





 ACTOR


 GAME


 Search








DirectionToServe real 


 Graphics



 Physics



 Motion



Linear Velocity

X

Y

Angular Velocity

Max Speed

Enforce Max Speed

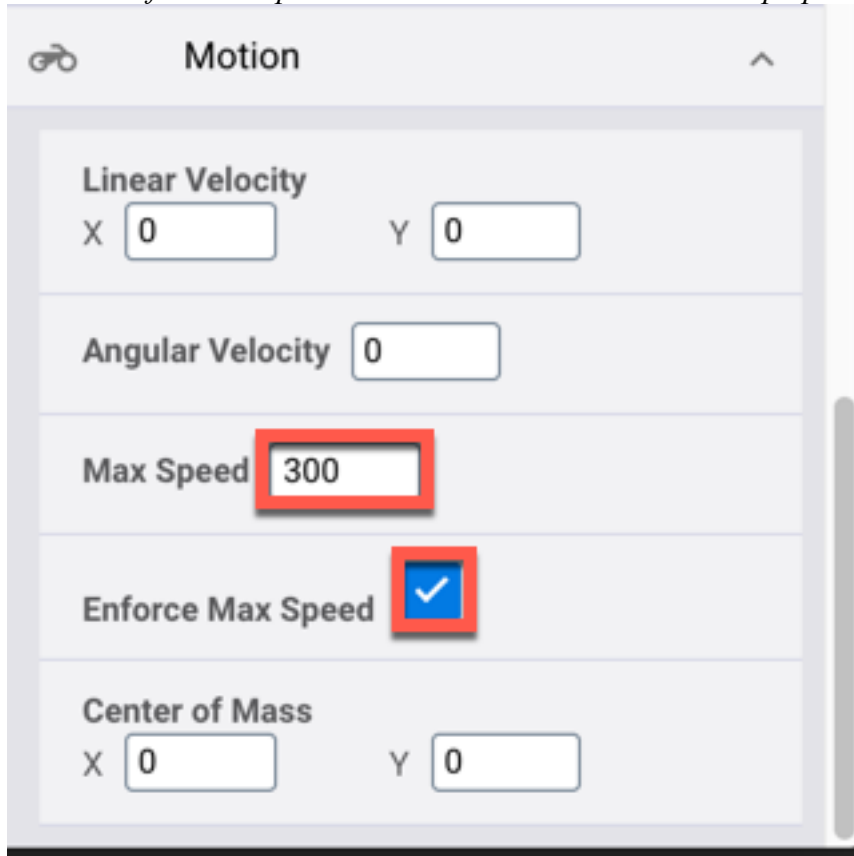
☐

Center of Mass

X

Y

16. Set the Max Speed attribute to 300 and turn on the Enforce Max Speed attribute. *(feel free to play around with different speed values, but keep in mind that the faster the ball, the faster the paddles should move to be able to keep up with it)*



Preview the game again and make sure the ball is the speed that you want it to be.

Unsticking the Ball

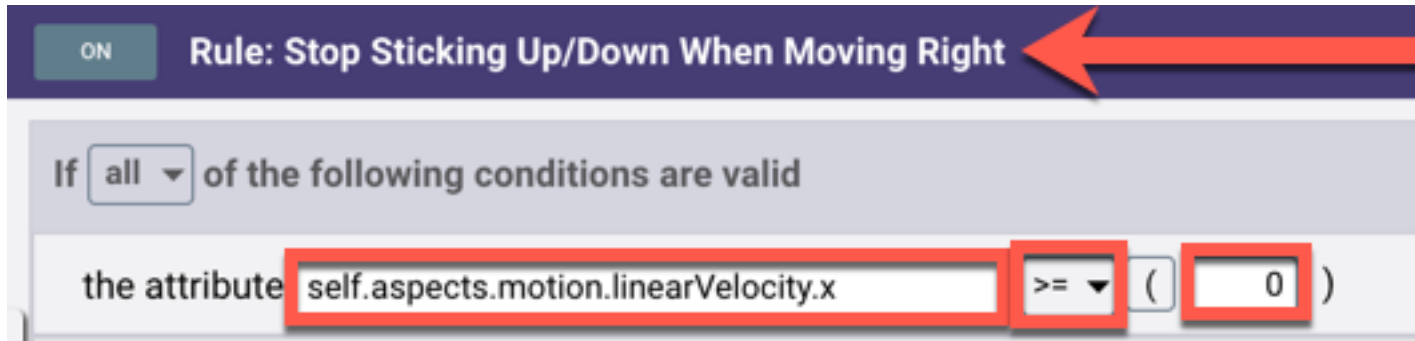
Since we are serving any random angle from 0-360, it's possible the ball will be serve straight up, down, left, or right, causing the ball to be stuck in an endless back and forth movement.

As a fun test, try turning off the change attribute behavior inside the ball actor that's setting the DirectionToServe, and change the attribute manually to 0, 90, 180, and 270 to see how it gets stuck.

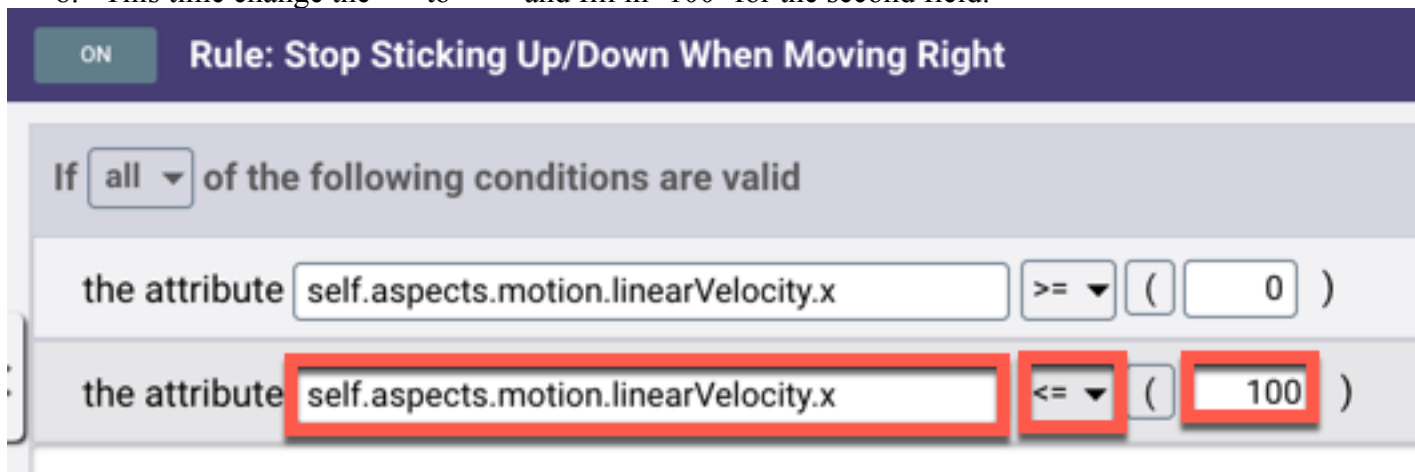
To fix that we can add some rules to check when the ball is in, or close to, that state, and give it a little nudge.

What attributes can we check to determine that the ball is in a stuck state? The linear velocity x and y attributes for the ball of course! When the linear velocity x (left/right motion) is close to 0, the ball will be bouncing straight up and down endlessly. The same goes for being stuck left/right when the linear velocity y attribute is close to 0.

1. Open the actor editor for the Ball actor.
2. Add a rule to the logic stack by clicking the blue 'Add Rule' button located in the Logic Stack Bar. Rename it "Rule: Stop Sticking Up/Down When Moving Right".
3. Delete the mouse pointer condition inside the rule and add an attribute comparison condition.
4. For the attribute field in the condition, select the attribute "self.motion.linearVelocity.x".
5. Change the '=' to '>='.
6. Fill in '0' for the second field.

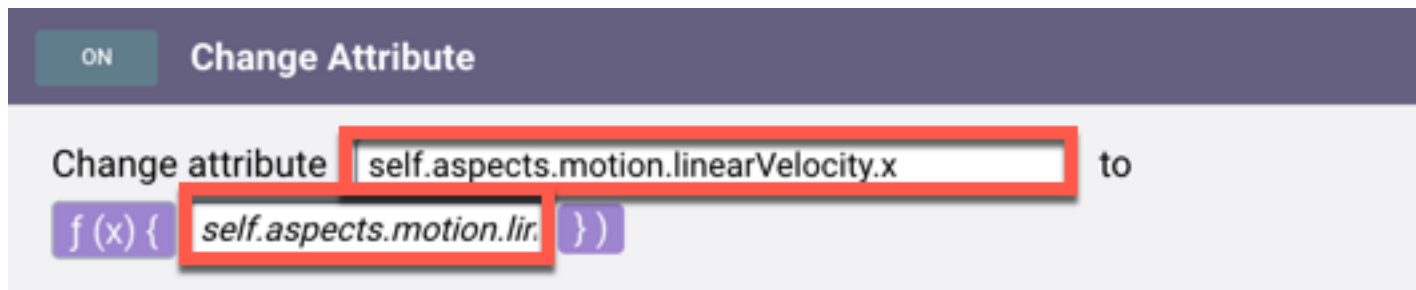


7. Add another attribute comparison condition and select the self.motion.linearVelocity.x attribute again.
8. This time change the '=' to '<=' and fill in '100' for the second field.



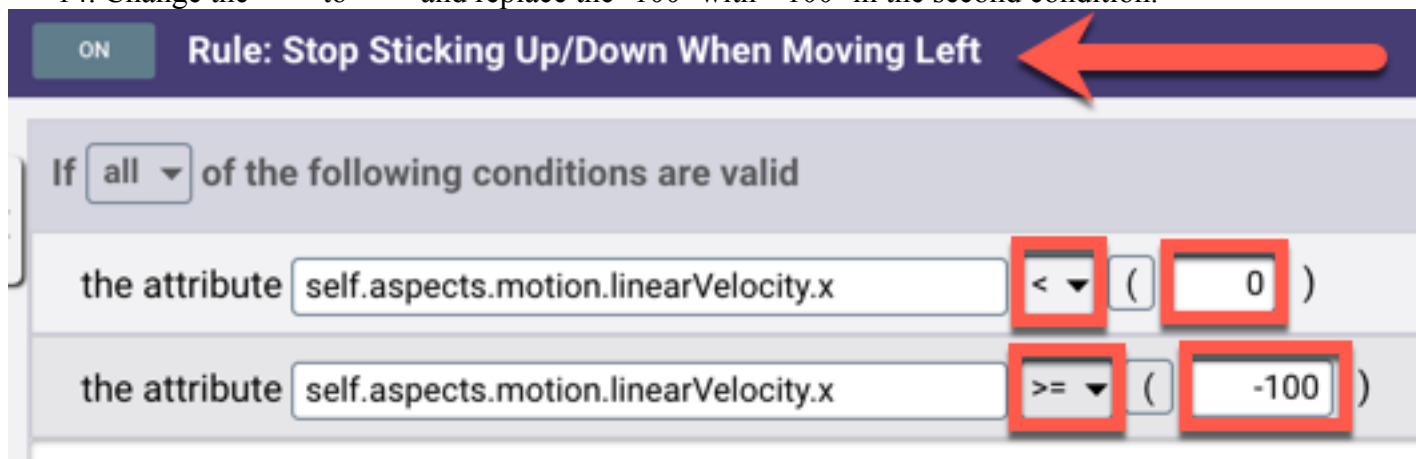
These conditions will ensure that the rule runs when the ball has a low velocity (between 0 and 100) while moving to the right on the screen.

9. Add a change attribute behavior inside the Stop Sticking rule.
10. Select the 'self.motion.linearVelocity.x' attribute for the first field and convert the second field to an expression / open the expression editor.
11. In the expression editor, fill in 'self.motion.linearVelocity.x + 100' and click the update button to save it. *(This will make sure that the new x velocity for the ball will be outside the conditions for the rule)*



We need another rule for when the ball has a low velocity, but is moving to the left.

12. Copy and paste the Stop Sticking rule you've made and rename it "Rule: Stop Sticking Up/Down When Moving Left".
13. Change the '>=' to '<' in the first condition.
14. Change the '<=' to '>=' and replace the '100' with '-100' in the second condition.



These conditions will ensure that the rule runs when the ball has a low velocity (between -1 and -100) while moving to the right on the screen.

15. Open the expression editor for the change attribute behavior inside the Stop Sticking when Moving Left rule and change it to "self.motion.linearVelocity.x-100".

We need two very similar rules to handle when the ball gets stuck bouncing back and forth left and right forever, but these will reference the self.linearVelocity.y attribute of the ball.

16. Add a new rule to the Logic Stack and rename it "Rule: Stop Sticking Left/Right When Moving Up".
17. Delete the mouse pointer condition inside the rule and add an attribute comparison condition.
18. For the attribute field in the condition, select the attribute "self.motion.linearVelocity.y".
19. Change the '=' to '<='.
20. Fill in '100' for the second field.
21. Add another attribute comparison condition and select the self.motion.linearVelocity.y

attribute again.

22. This time change the '=' to '>=' and fill in '0' for the second field.

ON Rule: Stop Sticking Left/Right When Moving Up

If of the following conditions are valid

the attribute ()

the attribute ()

Q Type down key to select condition or start typing to search

These conditions will ensure that the rule runs when the ball has a low velocity (between 0 and 100) in an upward direction.

23. Add a change attribute behavior inside the Stop Sticking rule.

24. Select the 'self.motion.linearVelocity.y' attribute for the first field and convert the second field to an expression / open the expression editor.

25. In the expression editor, fill in 'self.motion.linearVelocity.y + 100' and click the update button to save it. *(This will make sure that the new y velocity for the ball will be outside the conditions for the rule)*

ON Change Attribute

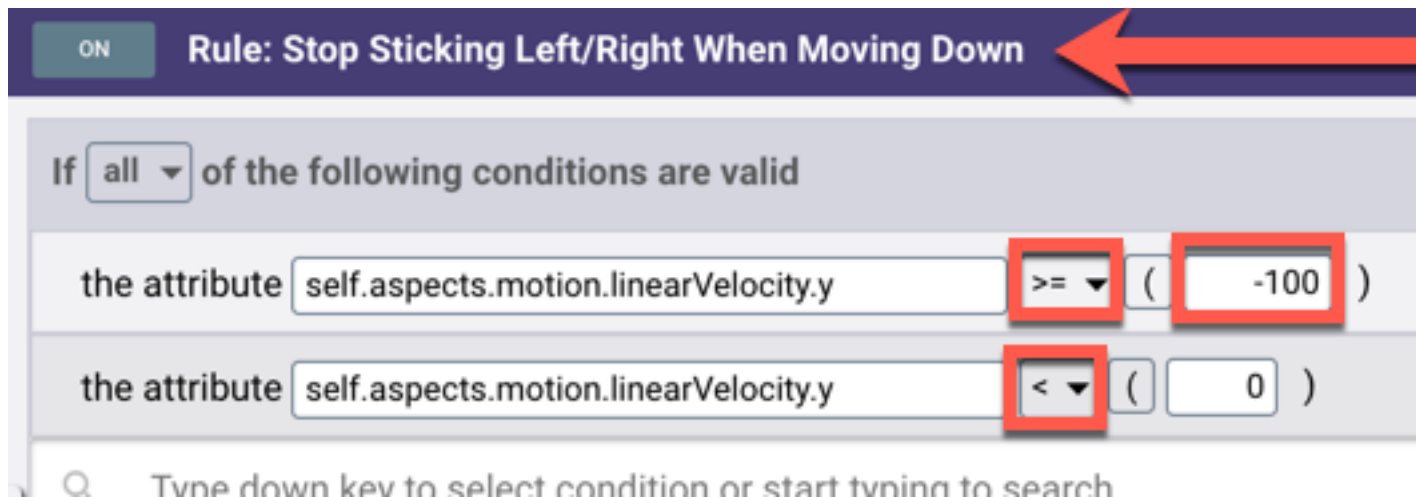
Change attribute to

We need another rule for when the ball has a low downward velocity as well.

26. Copy and paste the Stop Sticking rule you've made and rename it "Rule: Stop Sticking Left/Right When Moving Down".

27. Change the '<=' to '>=' and replace the '100' with '-100' in the first condition.

28. Change the '>=' to '<' in the second condition.



These conditions will ensure that the rule runs when the ball has a low downward velocity (between -1 and -100).

29. Open the expression editor for the change attribute behavior inside the Stop Sticking when Moving Down rule and change it to “self.motion.linearVelocity.y-100”.

Lastly, we need to place all the ‘Stop Sticking’ rules inside a timer that runs after 1 second. This is to prevent them from running while the ball is accelerating (when we serve it).

30. Minimize all the Stop Sticking rules.



31. Hold down shift and click on each Stop Sticking rule to highlight all of them.



32. Once they're all highlighted, click the timer button in the Logic Stack Bar to add a new timer to the Logic stack with the selected rules inside it.

The screenshot shows a configuration window for a timer. At the top, there is a dark blue header bar with a green 'ON' button and the title 'Timer'. Below the header, the configuration options are: a dropdown menu set to 'Every', a text input field containing '5', and the text 'seconds...'. To the right of these is an unchecked checkbox labeled 'Run to completion.'. Below this is a section titled 'Do' with a light blue background. On the left side of this section is a vertical scrollbar. Inside the 'Do' section, there are four dark blue bars, each containing a green 'ON' button and a rule name: 'Rule: Stop Sticking Up/Down When Moving Right', 'Rule: Stop Sticking Up/Down When Moving Left', 'Rule: Stop Sticking Left/Right When Moving Up', and 'Rule: Stop Sticking Left/Right When Moving Down'.

ON Timer

Every (5) seconds... ☐ Run to completion.

Do

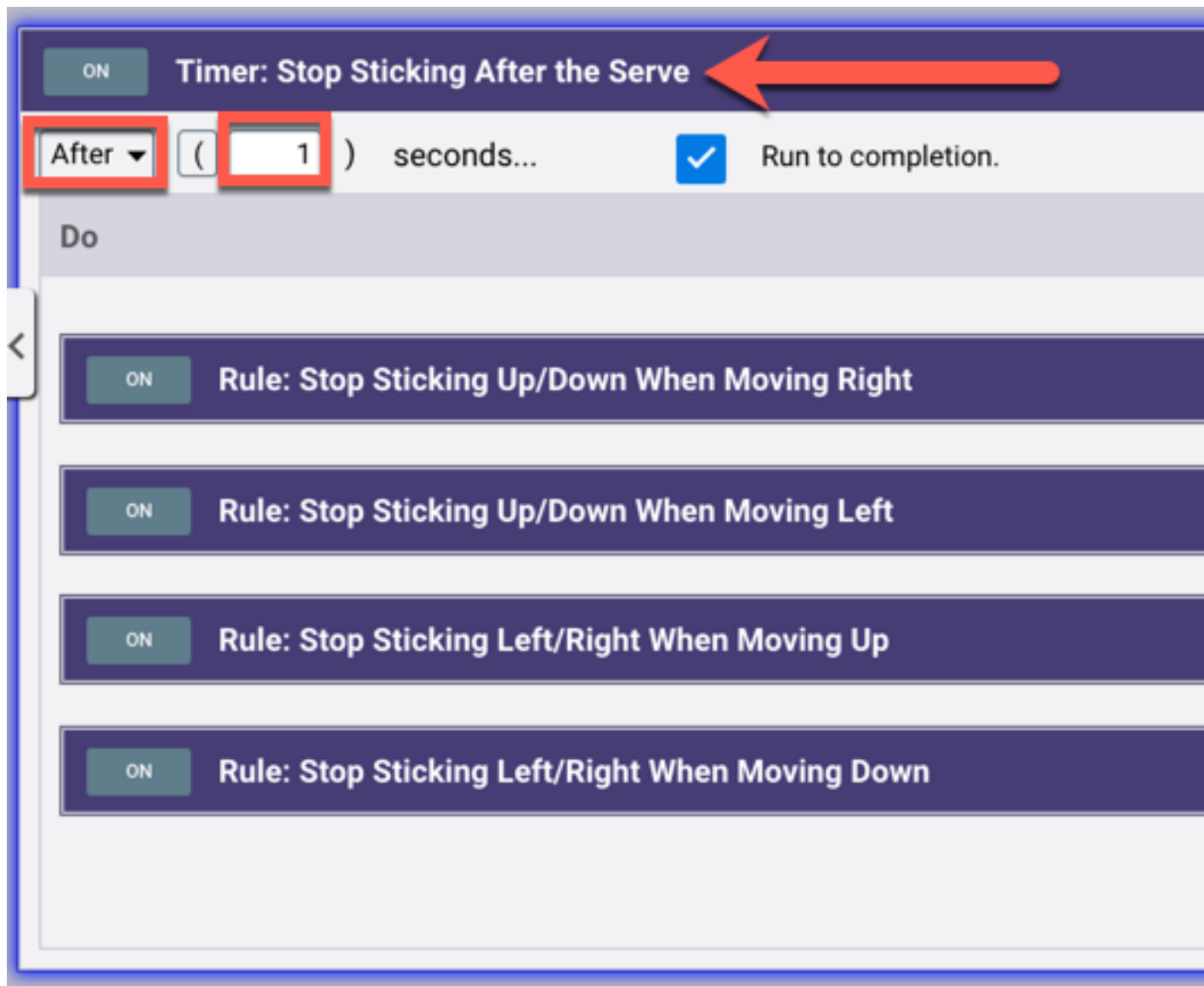
ON Rule: Stop Sticking Up/Down When Moving Right

ON Rule: Stop Sticking Up/Down When Moving Left

ON Rule: Stop Sticking Left/Right When Moving Up

ON Rule: Stop Sticking Left/Right When Moving Down

33. Rename it “Timer: Stop Sticking After the Serve”.
34. Change the ‘Every’ to ‘After’ and set the seconds value to ‘1’. Check the Run to Completion box.
35. It should look like this when you’re done.



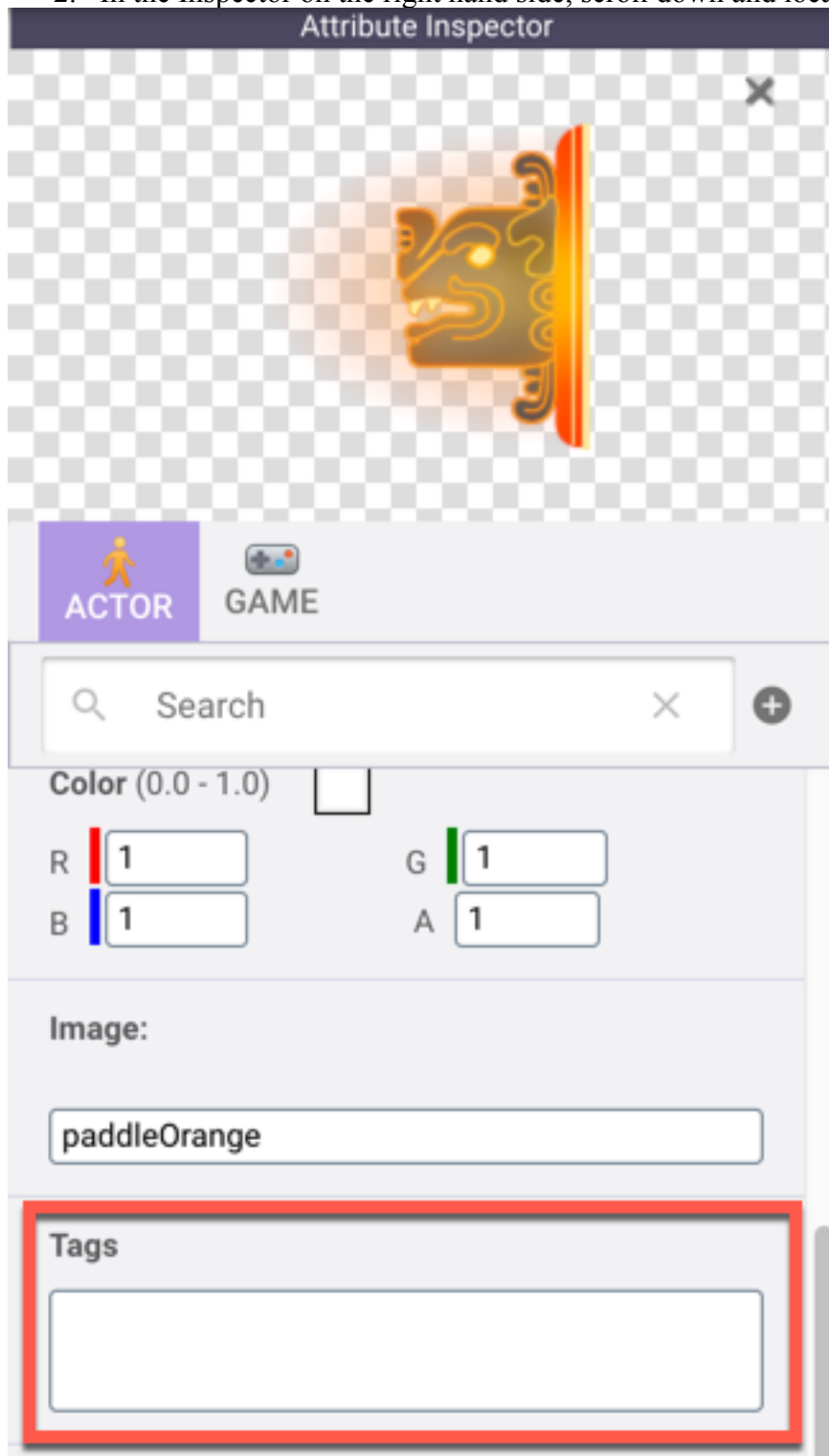
Try previewing the game after turning off the 'Choose Serve Direction' behavior inside the ball actor that's setting the DirectionToServe, and change the attribute manually to 0, 90, 180, and 270 to verify that it no longer gets stuck endlessly bouncing straight left/right or up/down. *(Be sure to turn the behavior back on when you're done!)*

Adding Polish

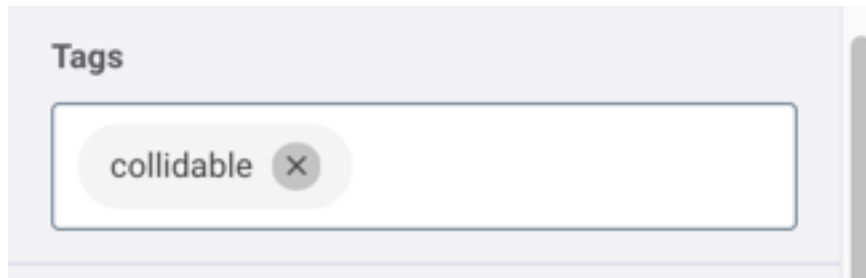
Sounds are an important part of any game and really bring them to life. Let's add some sounds to our project now!

1. Click on the Actors tab in the Library and select the LeftPaddle actor to navigate to the Actor Editor.

2. In the Inspector on the right hand side, scroll down and locate the Tags section.

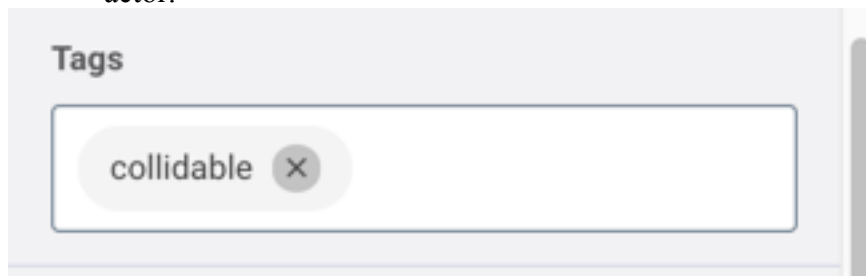


3. Click inside the Tags area and type “collidable” and hit Enter/Return on your keyboard.

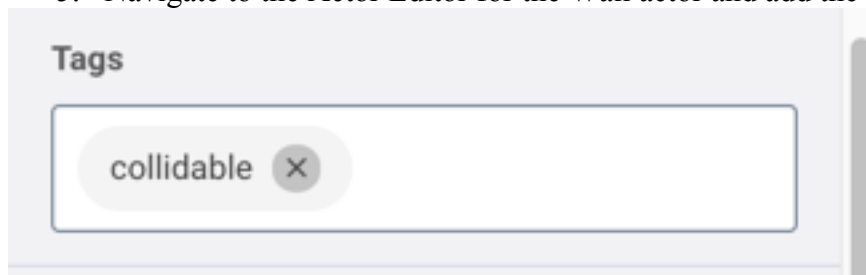


We need to add the collidable tag to our RightPaddle and Wall actors as well.


4. Navigate to the Actor Editor for the RightPaddle actor and add the collidable tag to the actor.



5. Navigate to the Actor Editor for the Wall actor and add the collidable tag to the actor.



6. Now that the collidable tag is applied to our actors, navigate to the Actor Editor for the Ball actor.
7. Create a new rule and rename it “Rule: Play Collision Sound”.
8. Change the mouse pointer condition to overlaps or collides.
9. Change the actor of type option to actor with tag.
10. Inside the field, type “collidable”.

ON Rule: Play Collision Sound 

If **all** of the following conditions are valid

this actor receives event **overlaps or collides** actor with tag **collidable**

Type down key to select condition or start typing to search.

Then

11. Add a play sound behavior to the rule.

12. Inside the Play Sound field, type “Hit_1” to add the sound to the behavior.

ON Play Sound

Play sound **Hit_1**

Play with Volume: (**1**) and Shift Pitch by (**1**)

Loop Sound: ☐ Run to completion: ☒

Positional Sound: ☐ Velocity Shift: ☐


When the ball collides with any of the actors that contain the “collidable” tag, a sound will play. Nice! Now let’s play another sound when one of the players score.

13. Navigate to the Actor Editor for the Goal for Left Paddle actor.

14. Create a new rule and rename it “Rule: Play Sound When Hit By Ball”.

15. Change the mouse pointer condition to overlaps or collides.

16. Inside the blank field, type “ball”.

ON Rule: Play Sound When Hit By Ball 

If **all** of the following conditions are valid

this actor receives event **overlaps or collides** actor of type **Ball**

🔍 Type down key to select condition or start typing to search.

17. Add a play sound behavior to the rule and play the “Chime32” sound.

ON Play Sound

Play sound **Chime32**

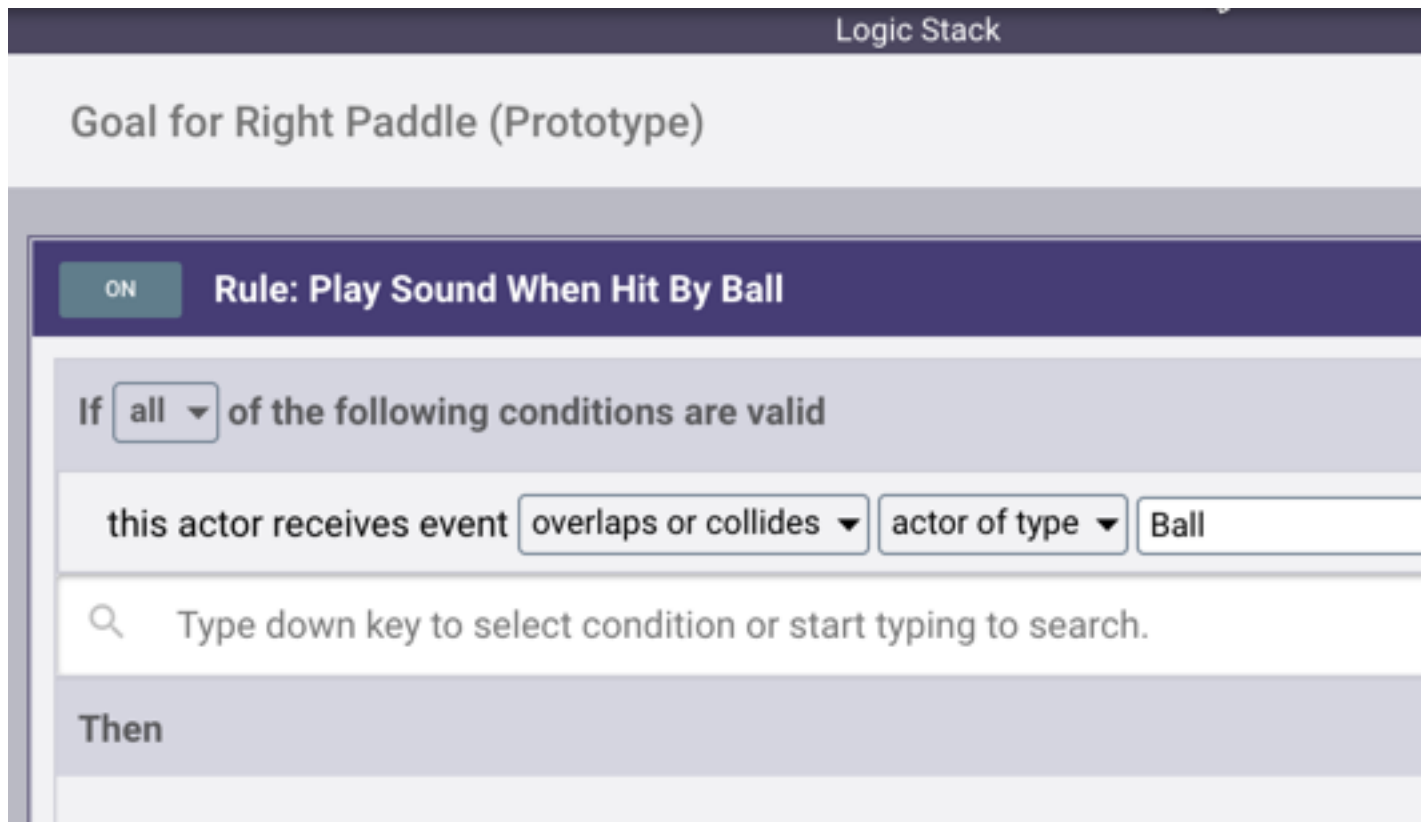
Play with Volume: () and Shift Pitch by ()

Loop Sound: ☐ Run to completion: ☒

Positional Sound: ☐ Velocity Shift: ☐

18. Copy the rule and navigate to the Actor Editor for the Goal for Right Paddle actor.

19. Paste the rule in the Logic Stack.



Critical thinking exercise for AI

How do you create an AI that's not impossible to beat?

How do you make one that's **also** not too easy to beat?

(I.e just constraining the paddle's y position to the y position of the ball would make it so that it never misses, and is thus impossible to beat, and making a paddle that always misses the ball would be too easy for the player.)