Adding sound effects

All sounds effects were created using Bfxr[1] and converted to an appropriate format with Audacity[2]. Start by adding the following code in the `mainState` object:

tf_scoreLeft

tf_scoreRight

sndBallHit

sndBallBounce

sndBallMissed

There are 3 sounds used in this game:

- A sound when the ball hits the paddle
- A sound when the ball bounces off the game world boundaries
- A sound when the ball goes outside our game world boundaries

We'll need a new function to initialise all our sound effects. After the `initKeyboard` function, add a new function called `initSounds`:

initKeyboardfunction

paddleLeft_upinputkeyboardaddKeyPhaserKeyboard

paddleLeft_downinputkeyboardaddKeyPhaserKeyboard

paddleRight_upinputkeyboardaddKeyPhaserKeyboard

paddleRight_downinputkeyboardaddKeyPhaserKeyboard

initSoundsfunction

This is where we will assign our game audio to the 3 sound properties created earlier. Add the following code:

initSoundsfunction

sndBallHitaudiosoundAssetsballHitName

sndBallBounceaudiosoundAssetsballBounceName

sndBallMissedaudiosoundAssetsballMissedName

Each sound is assigned to a property using a key or tag by calling the `game.add.audio` function. We assigned the keys for each sound in the `preload` function back in part 2[3] of this series.

Playing each sound

The first sound we will play is when the ball hits a paddle. Just add the following line of code in the `collideWithPaddle` function:

collideWithPaddlefunctionpaddle

sndBallHit

All that's needed to play a sound is to call the `play` function of a sound property.

Next, we'll add the sound for when the ball bounces off the game world boundaries. Add the following code to the `update` function:

updatefunction

moveLeftPaddle

moveRightPaddle

physicsarcadeoverlapballSpritepaddleGroupcollideWithPaddle

ballSpriteblockedballSpriteblockedballSpriteblockedballSpriteblockedright

sndBallBounce

At line 99, we are checking when the arcade physics body collides with any of the game world boundaries. When that happens, we play the bounce sound effect.

Our final sound effect plays when a player misses the ball and it goes outside the left or right game world boundaries. We add this in the `ballOutOfBounds` function:

ballOutOfBoundsfunction

sndBallMissed

Here is our current work in progress:

Click here[4] to download the source codes.

Adding the instructions and winner text fields

Let's add a new object called `labels` just before the `mainState` object:

```
labels

mainStatefunction
```

Next, we'll add 2 properties to the `labels` object:

```
labels

instructions'Left paddle: A to move up, Z to move down.nnRight paddle: UP and DOWN arrow keys.nn-
click to start -'

winner'Winner!'
```

The instructions text will only appear in demo mode while the winner text will appear on either the left or right side of the game world.

We'll also need to create a new font style for the text fields. Add the following code to the `fontAssets` object:

```
fontAssets

scoreLeft_xgamePropertiesscreenWidth

scoreRight_xgamePropertiesscreenWidth

scoreTop_y

scoreFontStyle'80px Arial''#FFFFFF'align'center'

instructionsFontStyle'24px Arial''#FFFFFF'align'center'
```

Also add the following 3 properties to the `mainState` object:

```
sndBallHit

sndBallBounce

sndBallMissed
```

instructions

winnerLeft

winnerRight

Now we'll create the text fields and place it within the game world. Add the following code to the `initGraphics` function:

tf_scoreRightfontAssetsscoreRight_xfontAssetsscoreTop_yfontAssetsscoreFontStyle

tf_scoreRightanchor

instructionsworldcenterXworldcenterYlabelsclickToStartfontAssetsinstructionsFontStyle

instructionsanchor

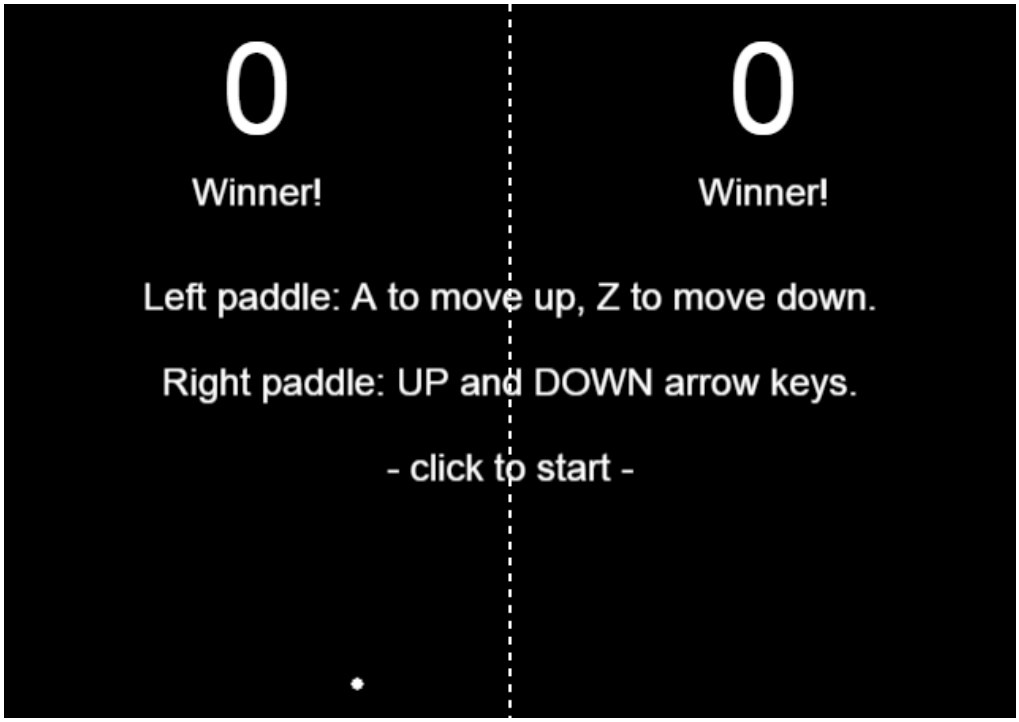
winnerLeftgamePropertiesscreenWidthgamePropertiesscreenHeightlabelswinnerfontAssetsinstructionsFontStyle

winnerLeftanchor

winnerRightgamePropertiesscreenWidthgamePropertiesscreenHeightlabelswinnerfontAssetsinstructionsFontStyle

winnerRightanchor

Here's a screen shot of what our current game should look like at this point:



Let's add a function called `hideTextFields` to hide all our text fields. We'll add it after the `updateScoreTextFields` function:

```
hideTextFieldsfunction

instructionsvisiblefalse

winnerLeftvisiblefalse

winnerRightvisiblefalse
```

We will call this function in the `initGraphics` function:

```
winnerRightgamePropertiesscreenWidthgamePropertiesscreenHeightlabelswinnerfontAssetsinstructionsFontStyle

winnerRightanchor

hideTextFields
```

The function will also need to be called in the `startGame` function:

```
startGamefunction

inputonDownremovestartGame

enablePaddles

enableBoundariesfalse

resetBall

resetScores

hideTextFields
```

Also, we will need to make the instructions text field visible in the `startDemo` function:

```
startDemofunction

ballSpritevisiblefalse

resetBall
```

enablePaddlesfalse

enableBoundaries

inputonDownstartGame

instructionsvisible

Next, let's update the `ballOutOfBounds` function to display the winner text field when either player hits the winning score:

updateScoreTextFields

scoreLeft>=gamePropertiesscoreToWin

winnerLeftvisible

startDemo

scoreRight>=gamePropertiesscoreToWin

winnerRightvisible

startDemo

resetBall

Notice we have split our single condition into two separate conditions to check if either side has reached the winning score.

Here is our current work in progress:

Click here[5] to download the source codes:

Wrapping up

To wrap up this series, let's add 3 more things:

1. Both paddles should appear at the vertical middle of the game world whenever a new game begins.
2. The original Pong had a small gap at the top of the game world so the paddles could not actually touch the very top.
3. After a few successful returns, let's increase the ball speed slightly to add to the game difficulty.

In the `enablePaddles` function, add the following code:

paddleLeft_upenabledenabled

paddleLeft_downenabledenabled

paddleRight_upenabledenabled

paddleRight_downenabledenabled

paddleLeftSpriteworldcenterY

paddleRightSpriteworldcenterY

We move both paddles y-position to the vertical centre of our game world.

Next, add another property called `paddleTopGap` to the `gameProperties` object:

paddleLeft_x

paddleRight_x

paddleVelocity

paddleSegmentsMax

paddleSegmentHeight

paddleSegmentAngle

paddleTopGap

This will be used in the `moveLeftPaddle` and `moveRightPaddle` functions. Add the following code in the `moveLeftPaddle` function:

paddleLeftSpritevelocity

paddleLeftSpritegamePropertiespaddleTopGap

paddleLeftSpritegamePropertiespaddleTopGap

Also add to the `moveRightPaddle` function:

paddleRightSpritevelocity

paddleRightSpritegamePropertiespaddleTopGap

paddleRightSpritegamePropertiespaddleTopGap

Lastly, we'll increase the ball movement speed after every 4 successful returns. Add the following code to the `gameProperties` object:

ballVelocity

ballRandomStartingAngleLeft

ballRandomStartingAngleRight

ballStartDelay

ballVelocityIncrement

ballReturnCount

We will also need to keep track of our current ball velocity in order to update it. In the `mainState` object, add the following property:

instructions

winnerLeft

winnerRight

ballVelocity

Every time the ball is reset, we will need to reset the ball velocity to its original value. This is done in the `startBall` function:

startBallfunction

ballVelocitygamePropertiesballVelocity

ballReturnCount

The `ballReturnCount` property (line 214) is used to keep track of how many times the ball has been returned.

In the `collideWithPaddle` function, we need to update one of the `velocityFromAngle` function arguments. Replace the `gameProperties.ballVelocity` argument with `this.ballVelocity`. Here is what the updated code looks like (lines 300 and 307):

paddlegamePropertiesscreenWidth

returnAnglesegmentHitgamePropertiespaddleSegmentAngle

physicsarcadevelocityFromAnglereturnAngleballVelocityballSpritevelocity

returnAnglesegmentHitgamePropertiespaddleSegmentAngle

returnAngle

returnAngle

physicsarcadevelocityFromAnglereturnAngleballVelocityballSpritevelocity

Finally, we will need add the following code to update the `ballReturnCount` and `ballVelocity` properties in the `collideWithPaddle` function:

ballReturnCount

ballReturnCount>=gamePropertiesballReturnCount

ballReturnCount

ballVelocitygamePropertiesballVelocityIncrement

Here's what our completed game looks like:

Click here[6] to download the finished game source codes.

Great work on completing this tutorial =) If you enjoyed this tutorial or know someone who might find it useful, feel free to share it with them. Also, do leave a comment and me some feedback or suggestions on other topics you would like to see on this website.

1. http://www.bfxr.net/
2. http://audacityteam.org/
3. http://zekechan.net/getting-started-html5-game-development-pong2/
4. https://github.com/zekechan/phaser-html5-tutorial-pong/releases/download/1.0/7a-Adding_sounds.zip
5. https://github.com/zekechan/phaser-html5-tutorial-pong/releases/download/1.0/7b-More_text_fields.zip
6. https://github.com/zekechan/phaser-html5-tutorial-pong/releases/download/1.0/7c-Wrapping_up.zip