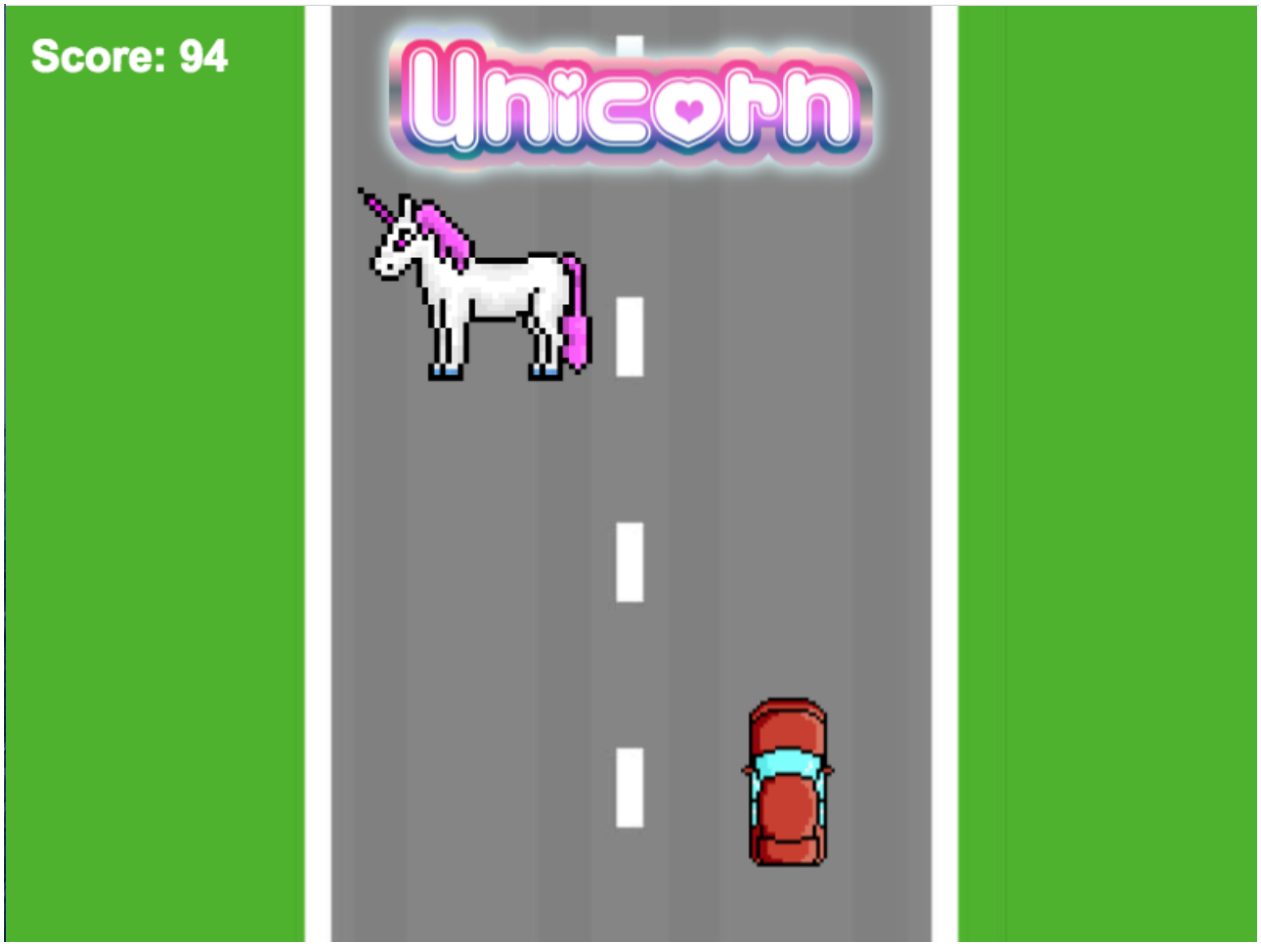
Donkey Corn

In this tutorial we are going to make a simple game using Phaser.io base on DONKEY.BAS a game written in 1981 and included with early versions of the PC DOS operating system distributed with the original IBM PC. It is a driving game in which the player must avoid hitting donkeys. The game was written by Microsoft co-founder Bill Gates and Neil Konzen.

In our updated version we are replacing Donkies with Unicorns and will have different mechanics for the car.



Pre Requisites

IDE VS Code, Python3 for a web server.

Source code.

You can find this tutorial on github.

https://github.com/dojojon/phaser/tree/master/donkey

Steps

All the assets for this tutorial along with the javascript library are in this directory. Create your game file in this folder as well.

Step 1

Make a copy of the step1.html file and name it game.html. You will add your code to this copy of the file as we go along.

Looking in this file you will see the following line of code. This sets up phaser, creating a variable called game that is 800 wide by 600 tall. We also set up functions that will get call by phaser that we can put our code in

```
    var game = new Phaser.Game(800, 600, Phaser.AUTO, '', { preload: preload, create: create, update: update,
render: render });
```

You can try running your game. To start the web server, run the following command in a terminal window.

```
python -m http.server
```

You should then be able to open the page in a web browser.

```
http://localhost:8000/game.html
```

You should see a black square in the browser.

Step 2

Next we need to load up our assets. To start with we just going to have images that we will use as sprites. In later steps we will add sound effects.

In the `preload()` function add the following:

```
        // Load the game assets
        game.load.image('road', 'assets/road.png');
        game.load.image('unicorn', 'assets/unicorn.png');
        game.load.image('car', 'assets/car.png');
```

To check everything is working lets add a sprite to the game. Below the line that creates the game add the following variable. This will allow us to reference the player object in other functions.

```
var player;
```

Next are going to add a player sprite. In the `create()` function

```
    player = game.add.sprite(game.world.centerX, game.world.height - 100, 'car');
    player.anchor.setTo(0.5, 0.5);
```

Save the file and reload the browser. Now we should see the players car displayed on the screen.

Step 3

Cool, we have a car but its floating in space. Lets add a road for it to drive on. For the road we are going to use a TileSprite. A TileSprite is a Sprite that has a repeating texture. Textures will automatically wrap and are designed so that you can create game backdrops using seamless textures as a source. Sounds perfect for our endless scrolling road.

Add the following to the `create()` function.

```
    //Set up road
    road = game.add.tileSprite(0, 0, 800, 20000, 'road');
    game.world.setBounds(0, 0, 800, 20000);
```

We will also tell phaser to always point the camera at the player. Below the code we used to create the player add the following;

```
game.camera.follow(player);
```

Save and run the game. You should see the car on top of the road.

Step 4

Lets get the car moving. We are going to use the build in arcade phyics engine. This will allow us set the speed (velocity) of spites in the game and detect when hit each other.

At the top of the create function add the following to enable the physics engine.

```
game.physics.startSystem(Phaser.Physics.Arcade);
```

Next we need to make the physics engine aware of our player sprite. Add the following line to the create function below the camera set up we added in the previous step.

```
game.physics.arcade.enable(player);
```

Next we need to add a variable for the cars speed. Up at the top of the file, add a variable called speed and set it to 200;

```
var speed = -200;
```

We can use this variable to set the speed of the player. In the update() function add the following.

```
player.body.velocity.y = speed;
```

Save and run the game. You should see the car on top of the road start to move.

Step 5

Next we going to the ability to controll the car using the cursor keys.

Add another variable to the top the file.

```
var cursors;
```

In the `create()` function we will add the code to set up the cursors.

```
cursors = game.input.keyboard.createCursorKeys();
```

To change the position of the car we can use an if statement to check to see if a cursor key pressed down. In the `update()` function add the following below the code that sets the players speed.

```
player.body.velocity.x = 0;

if (cursors.right.isDown) {
    player.body.velocity.x = 250;
}
if (cursors.left.isDown) {
    player.body.velocity.x = -250;
}
```

Save and run the game. You should be able to contol the car using the cursor keys.

Step 6

As you can image there is a lot going on in the game that is hidden from us. Each sprite has a position and a bunch of other properties that set define what it looks like and how it is drawn (render).

Phaser has some helper debug functions we can use to display this whilst the game is running. Lets see what the player debug info looks like.

In the `render()` function add the following.

```
game.debug.spriteInfo(player);
```

Save and run the game. You should see debug information displayed at the top of the screen. Try moving the can to see the x position change.

You probably don't want this information displayed all the time. So you can comment out the code using ```//```. That way it easy to add it back in when needed.

```
// game.debug.spriteInfo(player);
```

We can also print some text to the screen. Lets see if we can display the speed.

```
game.debug.text( "Speed " + speed , 100, 380 );
```

Step 7

We want our card to remain on the road. To do this we are going to create two invisible sprites to act as verges. We can the use the Arcade Physics engine to keep the car on the road.

Add another variable at the top of the code. This will be used to reference a group containing the two verges.

```
    var verges;
```

We will need to use a new asset, add the following to the preload.

```
game.load.image('pixel', 'assets/pixel.png');
game.load.image('clear_pixel', 'assets/transparent_pixel.png');
```

Next add a new function called under the `create()` function. We could put the code directly into the create function, but it helps when working on a larger project to group common things together into functions, even if they are only going to be called from one place.

```
function createVerges() {

}
```

We are going to create a group for the verges and enable physics for each sprite we put into it.

```
function createVerges() {

    verges = game.add.group();
    verges.enableBody = true;
    verges.physicsBodyType = Phaser.Physics.ARCADE;
}
```

Now create a sprite for the left side of the screen. We want this sprite to not move when hit by the car, so we set the immovable property to true. We need it to start at the top `(0 ,0)` corner of our world and stretch to the bottom of the world just before the road starts. We will do this by changing its size using scale and setting it to 200 wide and the height of the world.

```
function createVerges() {

    verges = game.add.group();
    verges.enableBody = true;
    verges.physicsBodyType = Phaser.Physics.ARCADE;

    var left = verges.create(0, 0, 'clear_pixel');
```

```
    left.body.immovable = true;
    left.scale.setTo(200, game.world.height);

}
```

Next add the right hand side.

```
function createVerges() {

    verges = game.add.group();
    verges.enableBody = true;
    verges.physicsBodyType = Phaser.Physics.ARCADE;

    var left = verges.create(0, 0, 'clear_pixel');
    left.body.immovable = true;
    left.scale.setTo(200, game.world.height);

    var right = verges.create(600, 0, 'clear_pixel');
    right.body.immovable = true;
    right.scale.setTo(200, game.world.height);

}
```

Your completed function should look like this.

```
function createVerges() {

    verges = game.add.group();
    verges.enableBody = true;
    verges.physicsBodyType = Phaser.Physics.ARCADE;

    var left = verges.create(0, 0, 'clear_pixel');
    left.body.immovable = true;
    left.scale.setTo(200, game.world.height);

    var right = verges.create(600, 0, 'clear_pixel');
    right.body.immovable = true;
    right.scale.setTo(200, game.world.height);

}
```

Now we need to call (run) this function from the `create()` function. You can call the function as follows.

```
createVerges();
```

The last step is to tell the physics engine to check for collisions on each update. Add the following to the update function.

```
// Player collides with verges
game.physics.arcade.collide(player, verges);
```

Save and run the game. The card should now remain on the road.

Step 8

Ok lets add the unicorns. dd another variable at the top of the code. This will be used to reference a group containing the unicorns.

```
    var unicorns;
```

Create another function to create the unicors, call it call it `createUnicorns()` .

```
function createUnicorns() {

}
```

Similar to the verges, we are going to create a group and set the physics on it.

```
function createUnicorns() {

    unicorns = game.add.group();
    unicorns.enableBody = true;
    unicorns.physicsBodyType = Phaser.Physics.ARCADE;

}
```

Next we a variable to hold the y position for each unicord we are going to add.

```
function createUnicorns() {

    unicorns = game.add.group();
    unicorns.enableBody = true;
    unicorns.physicsBodyType = Phaser.Physics.ARCADE;

    let posY = game.world.height;

}
```

We are going to use a for loop to create multipe unicorns. For loops are very handy. We can use then to loop round a number of times setting a variable to loop count. Below we are looping round 20 times and c will be set to the 0 all the way up to 19.

```
function createUnicorns() {

    unicorns = game.add.group();
    unicorns.enableBody = true;
    unicorns.physicsBodyType = Phaser.Physics.ARCADE;

    let posY = game.world.height;

    for (var c = 0; c < 20; c++) {

    }
}
```

Next we need to calculate the world position of the unicorn. So for each unicorn we add we will move it down the world (closer to the player) by 750 pixels.

```
function createUnicorns() {

    unicorns = game.add.group();
    unicorns.enableBody = true;
    unicorns.physicsBodyType = Phaser.Physics.ARCADE;

    let posY = game.world.height;

    // unicorn
    for (var c = 0; c < 20; c++) {

        //Set the next unicorn position
        posY = posY - 750;

    }
}
```

We want the unicorns to be either on the left or the right hand side of the road. We can use a build in Phaser function called chanceRoll that will return true or false randomly. By using the value 50 we have a 50% change of it being true or false.

```
function createUnicorns() {

    unicorns = game.add.group();
    unicorns.enableBody = true;
    unicorns.physicsBodyType = Phaser.Physics.ARCADE;

    let posY = game.world.height;
```

```
        // unicorn
        for (var c = 0; c < 20; c++) {

            //Set the next unicorn position
            posY = posY - 750;

            // Randomly position the unicorn
            const left = Phaser.Utils.chanceRoll(50);

        }
    }
```

Using the result of chance roll we can calculate the x position of the unicorn.

```
    function createUnicorns() {

        unicorns = game.add.group();
        unicorns.enableBody = true;
        unicorns.physicsBodyType = Phaser.Physics.ARCADE;

        let posY = game.world.height;

        // unicorn
        for (var c = 0; c < 20; c++) {

            //Set the next unicorn position
            posY = posY - 750;

            // Randomly position the unicorn
            const left = Phaser.Utils.chanceRoll(50);

            let posX = 0;
            if (left) {
                posX = game.world.centerX - 100;
            } else {
                posX = game.world.centerX + 100;
            }

        }
    }
```

Add last of all we can now create a unicorn in the unicorns group

```
    function createUnicorns() {

        unicorns = game.add.group();
        unicorns.enableBody = true;
        unicorns.physicsBodyType = Phaser.Physics.ARCADE;

        let posY = game.world.height;

        // unicorn
        for (var c = 0; c < 20; c++) {

            //Set the next unicorn position
            posY = posY - 750;

            // Randomly position the unicorn
            const left = Phaser.Utils.chanceRoll(50);

            let posX = 0;
            if (left) {
                posX = game.world.centerX - 100;
            } else {
                posX = game.world.centerX + 100;
            }

            // Create a unicorn
            var unicorn = unicorns.create(posX, posY, 'unicorn');
            unicorn.anchor.setTo(0.5, 0.5);

        }
    }
```

Now we need to call this function from the `create()` function. You can call the function as follows.

```
createUnicorns();
```

Save and run the game. We now have unicorns!!!!

Step 9

You may have noticed that we can drive through the unicorns. This is not them plan. Lets use the physics engine to detect it the car collides with a unicorn. Don't worry they are maigical creatures and can easliy survive this. As everyone know unicorns can disappear using rainbow magic so lets add some.

First we need to get the phaser physic engine to check for collisions between the player and the unicrons group. Add the following to the top of the `update()` funciton. Similar to the verges check, but this time we will call a function `collisionHandler` when a collision is deteced.

```
game.physics.arcade.overlap(unicorns, player, collisionHandler, null, this);
```

Lets add the function. We can put any code we want to run on a collision into this.

```
function collisionHandler(player, unicorn) {

}
```

First lets remove the unicorn from the game world.

```
function collisionHandler(player, unicorn) {

    unicorn.kill();

}
```

Next we will make the camera shake and flash the world.

```
function collisionHandler(player, unicorn) {

    // Kill the unicorn :-(
    unicorn.kill();

    // shake the screen and flash is white
    game.camera.shake(0.01);
    game.camera.flash();

}
```

Save the game and give it a play. Should should see the screen shake and flash along with the unicorn disappearing.

Setp 10

OK. We still don't have what we are looking for. Lets stop the player moving ona collision, its game over for them.

At the top of our script create a variable called `playing` and set this to true.

```
var playing = true;
```

In the `collisionHandler()` function set `playing` to `false`;

```
function collisionHandler(player, unicorn) {

    // Kill the unicorn :-(
    unicorn.kill();

    // shake the screen and flash is white
    game.camera.shake(0.01);
```

```
        game.camera.flash();

        playing = false;
    }
```

We can now use a if statement to check if this is true or false and change what code it run. If playing is false, we don't any of the code in the update function to be run, actually we just want the player to stay still in the world.

Coped below is the whole update function. As you can see we have moved all the current code inside the if statement and added an else to set the players speed to zero if playing is false.

```
function update() {

    if (playing) {

        // Player collides with verges
        game.physics.arcade.collide(player, verges);

        // Set the camera position so car is at bottom of the screen
        game.camera.focusOnXY(game.world.centerX, player.position.y - 200);

        // Check to see if we have hit any unicorns
        game.physics.arcade.overlap(unicorns, player, collisionHandler, null, this);

        //Set the players speed
        player.body.velocity.y = speed;

        player.body.velocity.x = 0;
        if (cursors.right.isDown) {
            player.body.velocity.x = 250;
        }
        if (cursors.left.isDown) {
            player.body.velocity.x = -250;
        }

        // Increase the player speed
        speed--;


    } else {

        // stop the player
        player.body.velocity.x = 0;
        player.body.velocity.y = 0;

    }

}
```

Save the game and run it. The player will now stop on a collision.

Step 11

Did I say rainbow magic. Lets add some using phasers built in particle emitters. Particles are a really easy way to add cool effects to your games. Take a look at the examples [https://phaser.io/examples/v2/category/particles]

First we need to add another asset. In the `preload()` function add the following;

```
game.load.image('rainbow', 'assets/rainbow.png');
```

Now lets add a function, that will take some coordinates as parameters and use a phaser emitter to make a bunch of rainbows. This means when we call this function we can pass through a position for the effect to be drawn at.

```
function rainbowExplode(posX, posY) {
}
```

Next we create an emitter for the rainbows.

```
function rainbowExplode(posX, posY) {

    // add a partical emitter
    emitter = game.add.emitter(posX, posY, 200);

    // make some particels
    emitter.makeParticles('rainbow');
}
```

The emitter can change the properties of the sprites it uses. Lets make rainbows of different sizes.

```
function rainbowExplode(posX, posY) {
    // add a partical emitter
    emitter = game.add.emitter(posX, posY, 200);

    // make some particels
    emitter.makeParticles('rainbow');

    // set the size range
    emitter.minParticleScale = 1;
    emitter.maxParticleScale = 3;
}
```

In a top down game we can set the gravity to zero.

```
function rainbowExplode(posX, posY) {
    // add a partical emitter
    emitter = game.add.emitter(posX, posY, 200);

    // make some particels
    emitter.makeParticles('rainbow');

    // set the size range
    emitter.minParticleScale = 1;
    emitter.maxParticleScale = 3;

    //turn off gravity
    emitter.gravity = 0;

}
```

We want the rainbows to fade away in 3 seconds, we can do this by setting the alpha

```
function rainbowExplode(posX, posY) {
    // add a partical emitter
    emitter = game.add.emitter(posX, posY, 200);

    // make some particels
    emitter.makeParticles('rainbow');

    // set the size range
    emitter.minParticleScale = 1;
    emitter.maxParticleScale = 3;

    //turn off gravity
    emitter.gravity = 0;

    emitter.setAlpha(1, 0, 3000);


}
```

Last of all, we need to tell phaser to do the emit.

```
function rainbowExplode(posX, posY) {
    // add a partical emitter
    emitter = game.add.emitter(posX, posY, 200);

    // make some particels
    emitter.makeParticles('rainbow');
```

```
        // set the size range
        emitter.minParticleScale = 1;
        emitter.maxParticleScale = 3;

        //turn off gravity
        emitter.gravity = 0;

        emitter.setAlpha(1, 0, 3000);

        // emit all the particles, for 3 seconds,
        emitter.start(true, 3000, null, 50);
    }
```

Add a call to this function in the `collisionHandler()` . We can pass through the players position, with a little offset to make explode happen at the front of the car.

```
    // Call the rainbow effect
    rainbowExplode(player.position.x, player.position.y - 50);
```

Save the game and give it a test.

Step 12

Ok. So we have a basic game, but it needs some tweaks to make it more enjoyable. First lets move the car towards the bottom of the screen. This will give the players more time to react to unicorns approaching.

In the `create()` function we set the camera up to follow the player. This means the car is in the center of the world.

Delete the following line from the create function.

```
    // Set the camera to follow the player
    game.camera.follow(player);
```

In the `update()` function add the following line to set the camera to the players position, but setting the y to be offset by -200.

```
    // Set the camera position so car is at bottom of the screen
    game.camera.focusOnXY(game.world.centerX, player.position.y - 200);
```

Save the game and give it a test. The car should now be at the bottom of the screen.

Step 13

Lets make the game get harder the longer it is played.

We can increase the speed in the `update()` function. Remember our game negative speed moves the player up the road.

```
speed--;
```

Each time this code is run it will add 1 to the speed. You can try other values using the following code.

```
speed = speed - 2.5;
```

Step 14

Game is not a game without a score.

Add two variables to the top of our script.

```
    // Contains player score
    var score = 0;
```

```
// Phaser text object for displaying score
var scoreText;
```

We are going to use Phasers text api to display the score. Add the following to the bottom of the create `function()`. We always want the score displayed on screen so we set the fixedToCamera to true.

```
// Set up score text
scoreText = game.add.text(16, 16, 'Score: 0', { fontSize: '32px', fill: '#FFF' });
scoreText.fixedToCamera = true;
scoreText.text = 'Score: ' + score;
```

Now we can add code to increase the score variable and upadte the text we display. Add the following to the top of the `update()` function to se the score text.

```
scoreText.text = "Score: " + score;
```

We only want to increase the score if the player is playing. Inside the `if(playing)` statement add the following.

```
score++;
```

Step 14

Lets add a little more polish to the game, a title. We are going to use a sprite and similar to the score lock this to the camera view.

Add a new variable at the top.

```
var logo;
```

We need to tell the phaser engine to load up the image. We do this by adding the following in the `prelaod()` function.

```
game.load.image('logo', 'assets/logo.png');
```

Next we need to load an image in the `create()` function. It positions it in center of the screen by dividing the camera width in 2 and 60 pixels from the top.

```
// Set up logo
logo = game.add.sprite(game.camera.width / 2, 60, 'logo');
logo.anchor.setTo(0.5, 0.5);
```

We want the logo to be fixed to the camera and not the world. So when the player moves we can still see it.

```
logo.fixedToCamera = true;
```

Save and run. You should see a logo at the top of the screen.

Step 15

Lets improve the game over by adding another sprite. This time we will add a tween (an animation) to it to make it more interesting.

Add a new variable at the top.

```
var gameoverlogo;
```

We need to tell the phaser engine to load up the image. We do this by adding the following in the `prelaod()` function.

```
game.load.image('game_over', 'assets/game_over.png');
```

You guess it, next we need to load an image in the `create()` function. This looks almost the same as the logo code, but we are positioning it in the center of the screen.

```
// Set up game over
gameoverlogo = game.add.sprite(game.camera.width / 2, game.camera.height / 2, 'game_over');
gameoverlogo.fixedToCamera = true;
gameoverlogo.anchor.setTo(0.5, 0.5);
```

We don't want the logo displayed until the player has crashed, so set the visibility flag to false.

```
gameoverlogo.visible = false;
```

We can use phasers build in tweening function to change properties on the sprite over time. The lines below change the scale (size) of the sprite from 1.0 to 1.05 and back every second. If you want to find out more, check out he phaser documentation. Add the following to the `create()` function.

```
game.add.tween(gameoverlogo.scale).to({ x: 1.05 }, 500, Phaser.Easing.Linear.None, true, 0, 1000, true);
game.add.tween(gameoverlogo.scale).to({ y: 1.05 }, 500, Phaser.Easing.Linear.None, true, 0, 1000, true);
```

Remember, we only want this displayed when the players game is over. We can do this in the `collisionHander()` fucntion that is called when we collide with a unicorn.

```
gameoverlogo.visible = false;
```

Save the game and run. Test out the game over sprite is displayed when we collide.

Step 16

Now its time to add some sound. Lets play a sound when the car collides with a unicorn. Sounds follow a similar pattern to sprites, define variable, load them, create an effect etc,

Add a variable at the top of the file.

```
var boing;
```

Next we need to load the sound effect. Add the following to the `preload()` function.

```
game.load.audio('boing', ['assets/boing.wav']);
```

And the same as images we need to to tell the phaser how to use then. Add the following to the `create()` function. We want the boing to be played only once, so we set loop to false.

```
boing = game.add.audio('boing');
boing.loop = false;
```

You probably guessed it, we want to play this sound when we collide with a unicorn. We can do this in the `collisionHander()` fucntion by adding the following.

```
boing.play();
```

Save the file and play the game to test it out.

Step 17

Lots of games use music, so lets add some to our game by using an mp3 file.

```
var music;
```

Next we need to load the mp3 effect. Add the following to the `preload()` function.

```
game.load.audio('flutes', ['assets/tkucza-happyflutes.mp3']);
```

Add the following to the `create()` function. We want the music to play in a loop, we will set the volume a little lower and start it off using the `play()` function.

```
music = game.add.audio('flutes');
music.loop = true;
music.volume = 0.5;
music.play();
```

Save the game and test it out.