CodeHS

Teacher Guide

Before the class:

- Make sure student computers have an up-to-date browser (Chrome, Safari, or Firefox).
- Read through teacher notes in this document. Download notes to have exercise solutions ready.
- Create a section with the JavaScript with Physics: Bouncing Ball Simulation

During the class:

1. Direct students to your section. If students need to create an account, help them create one prior to starting the class.
2. Allow students to work through the course at their own pace, providing encouragement and support when needed. See tips below for handling student questions.

Class Tips:

If students get stuck or have questions, it is okay if you don't have the answer! Ask questions to activate their problem-solving skills such as:

- What can we try differently?
- What do you want the program to do?

In this mini-unit, students will create a program that will apply your skills in creating a bouncing ball simulation, complete with gravity and collisions! Students should have some familiarity with physics concepts such as acceleration and velocity.

Objective

Students will be able to ...

- Utilize functions, parameters, graphics, and basic physics principles to create a bouncing ball simulation

Discussion Questions

- How do computer simulations of real life events help scientists?
- What are the benefits of using computer simulations?
- In what ways do computer generated values make scientists lives easier?

CodeHS

Teacher Guide

Exercise Solutions

Have A Ball	
Description	In this exercise you will get your "feet wet" by changing some properties of the ball. More precisely: - Its **radius** should be **20 pixels** - Its **starting height** should be t**wo thirds** from the bottom of the canvas. - You should also change the **color** of the ball to **red**. For **hints** on how to do this (and more) click on the **DOCS** tab. Remember that correct **capitalization** and **punctuation** (a.k.a. **syntax**) are extremely important in Javascript programming!
Motivation	This exercise allows students to practice with basic graphics functions to create a ball. We will build the animation in shortly.

Solution	`var RADIUS = 20;                        // In pixels`
	`var START_HEIGHT = 2 / 3 * getHeight(); // In pixels from bottom of canvas`
	
	`var ball;`
	
	`function start() {`
	`    ball = new Circle(RADIUS);`
	`    ball.setPosition(getWidth() / 2, getHeight() - START_HEIGHT - RADIUS);`
	`    ball.setColor(Color.RED);`
	`    add(ball);`
	`}`

Common Questions	**Misspelled commands or wrong letter case** In order for the program to run, every command must be spelled correctly and must be in the correct letter case. The command `setPosition` must be one connected command with the second word starting with an uppercase letter. **Nothing shows up on the screen** It is common for students to create the ball and set the position and color, but forget to use the add command.

To the Moon	
Description	Now things will get interesting because we are going to simulate how the ball would react to the force of gravity on the moon. Complete each of the following steps to witness this in action: 1. Do a bit of online research to determine the **acceleration due to gravity** of the moon and change the corresponding constant value in the code. How does

<table>
<tr><td></td><td>

changing it affect the time it takes for the ball to hit the ground?
2. Try changing the **frame rate** (FPS). How does changing the frame rate affect the simulation?
3. See if you can add a **WebImage** of a space capsule to the canvas right about where the ball would be dropped from. Check out the DOCS tab for information on how to do this. You can use the following url, or find one of your own: https://codehs.com/uploads/414e731d8a70863322abec26b6225458

</td></tr>
<tr><td>

Motivation

</td><td>

This exercise builds off the example allowing students to vary the effects of gravity.

</td></tr>
<tr><td>

Solution

</td><td>

```
var PIXELS_PER_METER = getHeight() / 3; // 3 meter canvas height
var GRAVITY = 1.62;                     // In meters/second^2
var FPS = 200;                          // Animation frames per second
var DELTA_TIME = 1 / FPS;               // In seconds

var RADIUS = 20;                        // In pixels
var START_HEIGHT = 2 / 3 * getHeight(); // In pixels from bottom of canvas

var ball;
var yVel = 0;

function start() {
    var lander = new WebImage("https://codehs.com/uploads/414e731d8a70863322abec26b6225458");
    lander.setSize(100, 128);
    lander.setPosition(getWidth() / 2 - 50, getHeight() - START_HEIGHT - 128 - RADIUS);
    add(lander);

    ball = new Circle(RADIUS);
    ball.setPosition(getWidth() / 2, getHeight() - START_HEIGHT - RADIUS);
    add(ball);
    setTimer(updateBall, DELTA_TIME * 1000);
}

/**
 * Update the location and velocity of the ball based on the frame rate.
 */
function updateBall() {
    yVel = yVel + GRAVITY * PIXELS_PER_METER * DELTA_TIME;
    var dY = yVel * DELTA_TIME; // Calculate change in y position
    ball.move(0, dY);   // Move the ball relative to current position
}
```

</td></tr>
<tr><td>

Common Questions

</td><td>

How do I change gravity?
Since we use a constant, you only need to change the gravity variable at the top of the

</td></tr>
</table>

program.

How do I find the value of gravity?
Look only. You should be able to find the value of gravity on the moon expressed in meters per second squared.

Off the Wall!

Description

In this exercise, your task is to add a bit of **initial horizontal velocity** to the ball and cause it to bounce off of the walls.

You will need to rely on the following equation to simulate the **reflection** of the ball in the x axis:
`velocity = -velocity`

Motivation

This exercise builds on the example and provides most of the code the students need. They will need to understand how the code works so that they can make the adjustments needed.

Solution

```
/**
 * This program adds horizontal velocity and collision
 * detection to our bouncing ball simulation.
 */

var PIXELS_PER_METER = getHeight() / 3; // 3 meter canvas height
var GRAVITY = 9.8;                      // In meters/second^2
var FPS = 100;                          // Animation frames per second
var DELTA_TIME = 1 / FPS;               // In seconds

var RADIUS = 20;                        // In pixels
var START_HEIGHT = 2 / 3 * getHeight(); // In pixels from bottom of canvas

var ball;
var yVel = 0;
var xVel = 300;

function start() {
    ball = new Circle(RADIUS);
    ball.setPosition(getWidth() / 2, getHeight() - START_HEIGHT - RADIUS);
    add(ball);
    setTimer(updateBall, DELTA_TIME * 1000);
}
```

```
/**
 * Update the location and velocity of the ball.
 * Check for collisions with the ground and side walls.
 */
function updateBall() {
    if (ball.getY() + RADIUS >= getHeight()) {
        yVel = -yVel;   // Simulate collision with the ground
    } else {
        yVel = yVel + GRAVITY * PIXELS_PER_METER * DELTA_TIME;
    }
    if (ball.getX() + RADIUS >= getWidth()
        || ball.getX() - RADIUS <= 0) {
        xVel = -xVel;
    }
    var dY = yVel * DELTA_TIME; // Calculate change in y position
    var dX = xVel * DELTA_TIME; // Calculate change in x position
    ball.move(dX, dY);   // Move the ball relative to current position
}
```

Common Questions	**I updated the xVel but nothing happened** In addition to updating the velocity, you also need to change the ball.move command to take the dX value. **Why doesn't my ball move very fast?** Make sure you update the xVel so that it is greater than 10, otherwise the ball movement will be pretty small.

Keepin' it Real (Elastic)	
Description	Now that you're mastering what it takes to create a simulation, update your program to simulate **elastic collisions** both with the ground and the wall! In an elastic collision a bit of the **kinetic energy** of the ball is converted to heat and so it doesn't bounce back with the same velocity. This can be simulated by multiplying the velocity of the ball with an **elasticity coefficient** when the ball collides with a surface: `velocity = -velocity * elasticityCoefficient;` Where the elasticity coefficient is a decimal value less than 1 (0.9 is a good start). Some additional features you might want to add to the final version include: ● Playing a sound when the ball hits the wall ● Adding a friction coefficient for the ground (for when the ball's y position is less than or equal to zero) ● Adding a friction coefficient for the air (will always slow the ball down in all

<table>
<tr><td></td><td>directions by some amount).</td></tr>
<tr><td><b>Motivation</b></td><td>This final challenge has students adding an elasticity coefficient to their program to help make the ball bounce in a more realistic manner.</td></tr>
<tr><td><b>Solution</b></td><td>

```
/var PIXELS_PER_METER = getHeight() / 3; // 3 meter canvas height
var GRAVITY = 9.8;                      // In meters/second^2
var FPS = 100;                          // Animation frames per second
var DELTA_TIME = 1 / FPS;               // In seconds

var RADIUS = 20;                        // In pixels
var START_HEIGHT = 2 / 3 * getHeight(); // In pixels from bottom of canvas

var ELASTICITY = 0.9;

var ball;
var yVel = 0;
var xVel = 200;

function start() {
    ball = new Circle(RADIUS);
    ball.setPosition(getWidth() / 2, getHeight() - START_HEIGHT - RADIUS);
    add(ball);
    setTimer(updateBall, DELTA_TIME * 1000);
}

/**
 * Update the location and velocity of the ball.
 * Check for collisions with the ground and side walls.
 */
function updateBall() {
    if (yVel > 0 && ball.getY() + RADIUS >= getHeight()) {
        yVel = -yVel * ELASTICITY;   // Simulate collision with the ground
    } else {
        yVel = yVel + GRAVITY * PIXELS_PER_METER * DELTA_TIME;
    }
    if ((xVel > 0 && ball.getX() + RADIUS >= getWidth())
        || (xVel < 0 && ball.getX() - RADIUS <= 0)) {
        xVel = -xVel * ELASTICITY;
    }
    var dY = yVel * DELTA_TIME; // Calculate change in y position
    var dX = xVel * DELTA_TIME; // Calculate change in x position
    ball.move(dX, dY);   // Move the ball relative to current position
}
```

</td></tr>
</table>

Teacher Guide

Common Questions	**My ball seems to slow down too quickly.** Make sure you are applying the elasticity in the correct places. It should only be applied when the ball hits an object, such as the floor or wall.