SUPINFO Academic Dept.

SUPINFO International University

INSTITUTE OF INFORMATION TECHNOLOGY

BreakDown
A.Sc.1 - Associate of Science 1

Contents

Version 1.0
Last update: 2017-12-05
Use: Students
Author: Renaud BERTHIER

TABLE OF CONTENTS

1. BreakDown

1.1. Context

You sure know Breakout! If not, check the following link: https://en.wikipedia.org/wiki/Breakout_(video_game)

Gaming Company asked you to develop a web version of Breakout. But as they want the project to work on very old versions of Internet Explorer and as they hate polyfills for some reasons, you must develop this game **without Canvas, and without any third-party libraries (at the sole exception of jQuery). Any jQuery plugin is also forbidden (even jQuery UI).**

1.2. What's expected

With only HTML, CSS and JS, create a Breakout game with the following features:

- Create a game area
- Generate random blocks inside the top 50% of the game area
- Move the bar according to the mouse movement
- Move the ball in a understandable way
- The further from the bar center the ball hits, the more inclined the path should be
- Bonus items might appear randomly after breaking a brick
- Handle collisions between the ball and the bar, the game area or a brick, and between the bar and the items
- The game is won when no brick is left in the game area
- The game is lost if no more ball is shown on the game area

1.2.1. Create a game area

Your game area should be at first 650px wide and 600px high. Note that this size should be easily changed within the code without interface problems. Position this element as relative and in the middle of the window.

Your application should contain a lot of variables to handle every element of the game: Bar dimensions, ball size, Bricks dimensions, Game Area dimensions, Bonus dimensions and effects... See below for more information.

1.2.2. Generate random blocks

In the top 50% of the game area, bricks should be displayed. Bricks belong to an invisible grid and such cannot be placed outside this grid.

For each cell in this grid, a brick might or not be generated based on random. Create a random color (with RGB) for every brick.

1.2.3. Move the bar

When the user moves the mouse all over the window, the bar should move on its X-axis following the mouse movement. Of course, the bar should not get out of the game area.

1.2.4. Move the ball

Keep track of the angle of the ball when it's moving. When the ball hits a vertical wall, its horizontal direction should reverse. When the ball hits the top of the game area, its vertical direction should reverse. And when the ball hits the bottom of the game area, the game is lost. More information on the dedicated section.

1.2.5. The further the touch, the inclined the path

Implement a formula in which:

- When the ball hits the bar in its middle, the ball should bounce vertically
- When the ball hits the bar in its extreme, the ball should bounce with an angle roughly equal to 75°. The important here is not the value but the gameplay.

Be careful not to create a horizontal bounce, which would lead in a never ending game.

1.2.6. Bonus items

When breaking a brick, a bonus item might appear. See the list of bonuses you need to implement (feel free to add more):

- Green item: Must add 10px to the bar (max 50% of the game area width)
- Red item: Must remove 10px to the bar (min twice the ball size)
- Blue item: Must cast another ball in the game area

The bonus when appearing is falling down to the bottom of the game area. To catch a bonus, the bar should collide with the bonus. If the bonus is missed, nothing happens. On the contrary, the bonus effect should be applied.

When having multiballs, fail to catch a ball won't mean a defeat ; the ball is removed from the game area.

1.2.7. Handle collisions

Of course you will need to handle collisions. Collisions are usually a hard part in game development, so visually perfect collisions are not mandatory. However, your elements must behave logically.

- When colliding a brick on a side, the ball should go the opposite way.

- The bar should only move on the X-axis, and never move outside of the game area.
- Balls are strictly confined in the game area.
- Bricks should not move and be breakable.

1.2.8. Win conditions

The game is won when no brick are left on the game area. To play a new game, just refresh the page.

1.2.9. Lose conditions

The game is lost when no ball is left on the game area. To play a new game, just refresh the game.

1.3. What you need to do/know

1.3.1. Understand Client Need

Before you start, make sure you've understood the client's needs and have thought the project out. Write stuff down, draw schemas if you need to.

THINK SIMPLE! DO NOT GO OUT OF YOUR WAY IN COMPLEXITY.

1.3.2. A little help

The Mozilla Developer Network documentation is a very well-documented encyclopedia with lots of code and examples.

1.3.3. A Quick Reminder

Courseworks have a research phase. Some of the tasks required may not be listed in your lessons, you will probably need to do a bit of research.

N.B: Looking something up on Google is not copying and pasting code. Read, understand, and do your own version according to your needs. Copy-pasting will be considered as cheating and stealing.

1.4. Evaluation

1.4.1. No cheating

This is an individual project, therefore you are not allowed to host your project online, use someone else's code, copy and paste code from the internet. If someone is caught cheating, he will receive a 0 mark and be marked as cheater.

1.4.2. Grading Scale

- Game Area, Brick generation, Bar generation (3 points)

- The ball is moving in a convenient way (4 points)
- Bonus are working perfectly (5 points)
- Handle collisions (6 points)
- Win and lose conditions (3 points)
- User Interface and Design (3 points)
- Code Quality and Conventions (3 points)