BASECASE

BaseCase Defender Programming Test

Thank you for your interest BaseCase! We hope you enjoy tackling this programming test.

This is not specifically a front-end developer test.

This test requires only writing the logic for the movement of the paddle and not the UI.

You must write a solution to this test using Javascript, but note that you do not need to interact with the 'document object model' (DOM), and so this test is not specifically a 'front-end' test. Even if you have limited understanding of Javascript, you should be able to tackle this test.

For those of you new to Javascript, I would suggest review the math and array functions available in Javascript:

- http://www.w3schools.com/jsref/jsref_obj_math.asp
- http://www.w3schools.com/js/js_arrays.asp

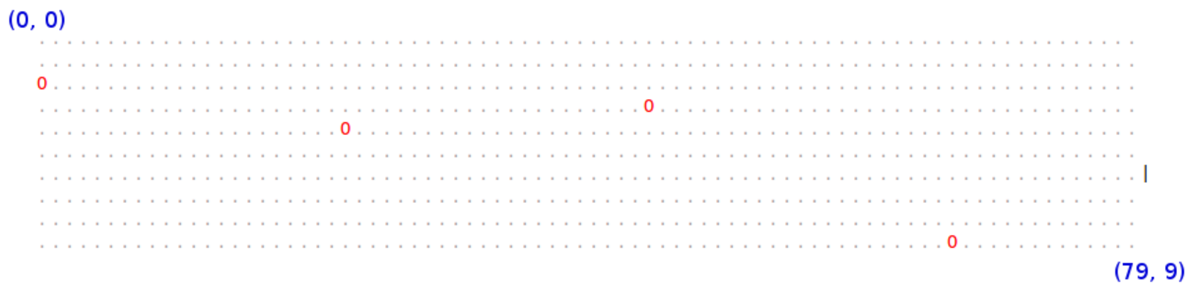
Contents

Introduction 3
Implementing the AI 4
Game Loop 5
The Game Loop in pseudo code: 5
How to build your solution in JS FIddle 6
How to submit your JS Fiddle solution 7
Grading your Solution 8

Introduction

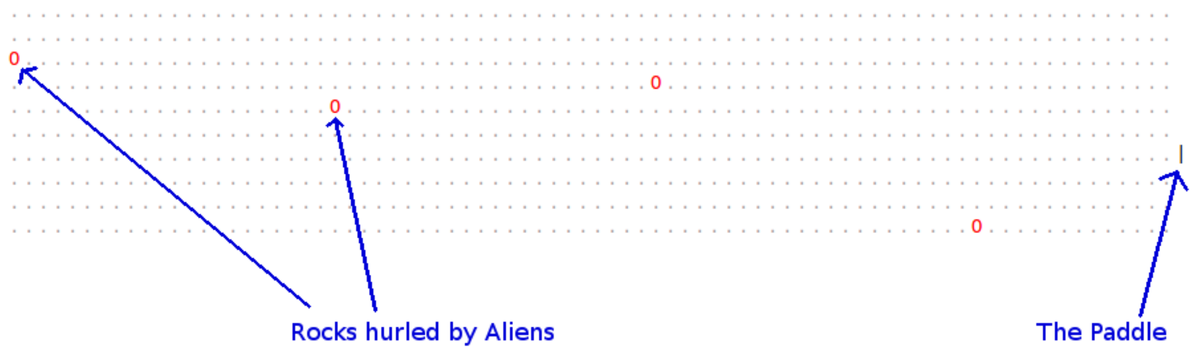
The Earth is under attack! You must write an AI to protect the Earth from an onslaught of red rocks.

The action takes place on an 80x10 sensor grid:



Note that **(x=0, y=0)** is the top-left, and **(x=79, y=9)** is the bottom-right.

On the sensor grid you will see rocks hurling towards the Earth (which is itself off-screen somewhere to the right). The only thing that stands between the Earth and total annihilation is **the Paddle**:



The Paddle is located along the **x=80** axis, and can only move 'up' and 'down'. You will need to create an AI for The Paddle so that it intercepts the rocks and saves the Earth.

NB: Due to technical limitations in Alien technology, the rocks are always 21 grid points apart.

Implementing the AI

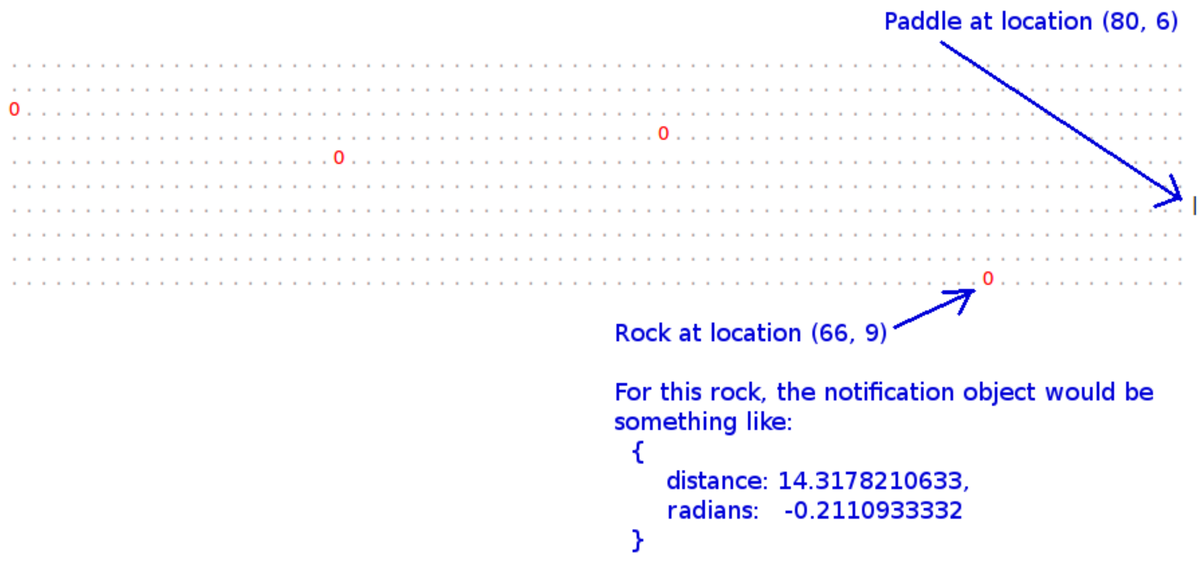
The game updates every ~100ms. For each update, the rocks advance one grid point to the right, and the Paddle can do one of the following: move up one grid point, move down one grid point, stay where it is.

To save the Earth, you must implement the AI for the Paddle. This is done via the 'notify_player()' notification function that is called every time a rock enters or leaves the sensor grid:

```
defender.start(function notify_player(rocks, paddle_y){

        var moves = [];

        // compute your moves

        // ...

        return moves;

} );
```

Notes on parameters to this function:

- The 'rocks' variable contains the list of rocks visible in the sensor grid. They are in no particular order, and each rock notification object contains the distance from the rock to the Paddle, and the angle (in radians) of the line connecting the rock to the Paddle.
- The 'paddle_y' variable function is the current y-location of the Paddle.


Paddle at location (80, 6)
Rock at location (66, 9)
For this rock, the notification object would be something like:
{
distance: 14.3178210633,
radians: -0.2110933332
}

The 'moves' array returned by the 'notify_player()' function specify the moves you would like your paddle to make in the subsequent updates. A valid move is one of -1 (move up), 0 (stay where you are), and 1 (move down). So, returning [-1, 1, 0] means "move up, then move down, then stay where you are". The 'moves' returned will overwrite any previous list of moves provided by the AI in an earlier invocation of notify_player().

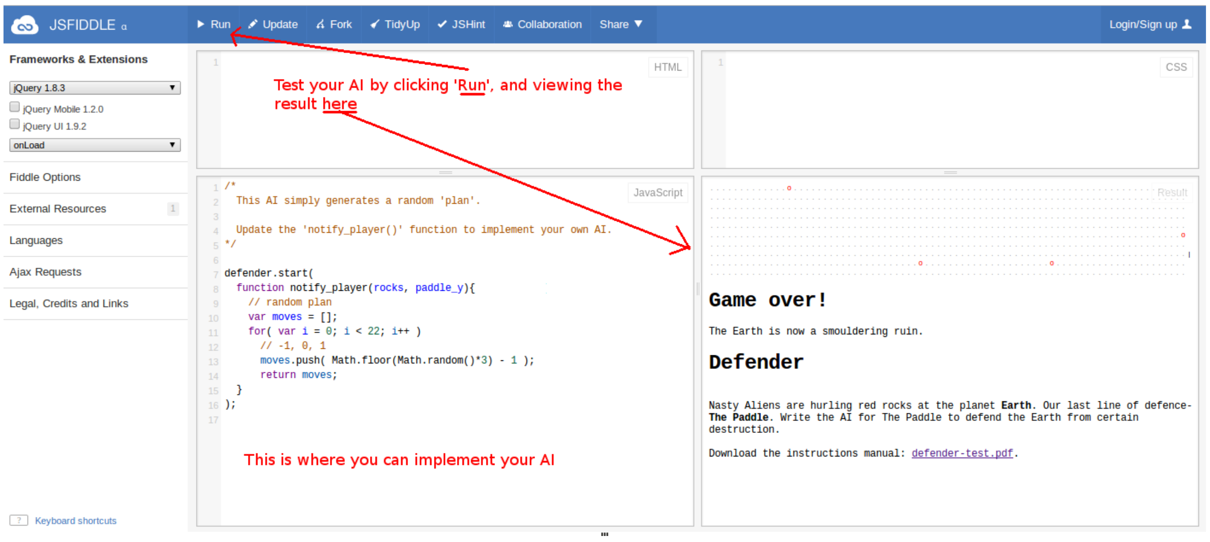
Game Loop

The Game Loop in pseudo code:

```
1. moves = []
2. Loop:
    a. Move each rock 1 grid point to the right
    b. If a rock is at x=80
        i. If the rock has not been intercepted by the Paddle, then Game
           Over
       ii. Otherwise, remove the rock from the sensor grid
    c. If there are no rocks in the region (x=0..20), then add a new rock
       along the x=0 axis.
    d. If a rock has been added to / removed from the sensor grid, call the
       notify_player(rocks, paddle_y), function, storing the return value in
       the 'moves' variable
    e. If len(moves) > 0
        i. Then next_move = moves.pop()
       ii. Otherwise next_move = 0
    f. Paddle.Y += next_move
```

How to build your solution in JS FIddle

As part of this test, you should have received a URL to JS Fiddle. You can create and test your AI in the JS Fiddle interface:



Test your AI by clicking 'Run', and viewing the result here

```
/*
  This AI simply generates a random 'plan'.

  Update the 'notify_player()' function to implement your own AI.
*/

defender.start(
  function notify_player(rocks, paddle_y){
    // random plan
    var moves = [];
    for( var i = 0; i < 22; i++ )
      // -1, 0, 1
      moves.push( Math.floor(Math.random()*3) - 1 );
      return moves;
  }
);
```

This is where you can implement your AI

Game over!

The Earth is now a smouldering ruin.

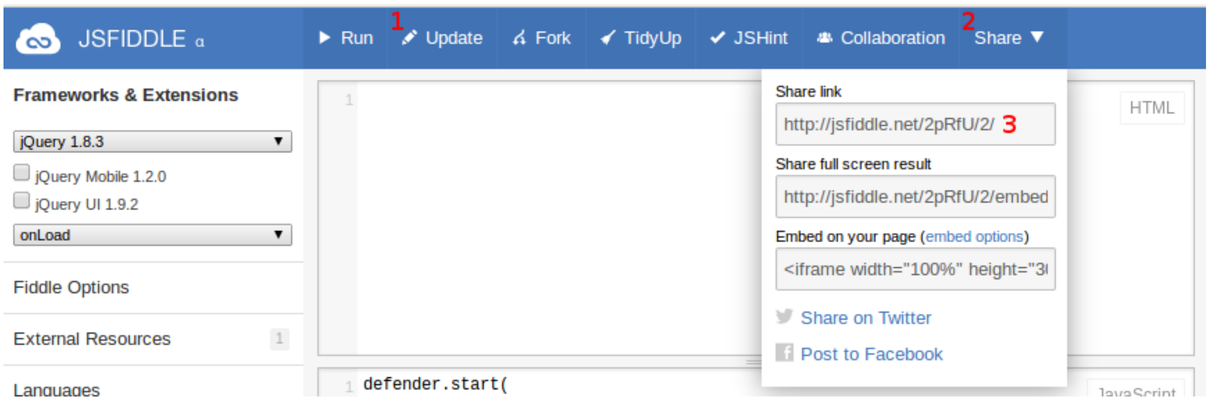
Defender

Nasty Aliens are hurling red rocks at the planet **Earth**. Our last line of defence- **The Paddle**. Write the AI for The Paddle to defend the Earth from certain destruction.

Download the instructions manual: defender-test.pdf.

How to submit your JS Fiddle solution

When you are ready to submit your solution:



jQuery 1.8.3

☐ jQuery Mobile 1.2.0

☐ jQuery UI 1.9.2

onLoad

External Resources 1

```
defender.start(
```

Share link: http://jsfiddle.net/2pRfU/2/ 3

Share full screen result: http://jsfiddle.net/2pRfU/2/embed

Embed on your page (embed options): <iframe width="100%" height="3

1. Click ‘Update’
2. Click ‘Share’
3. Email us the ‘Share link’ (preferably by replying to the email that sent you this test – otherwise send your solution to careers@basecase.com).

To confirm that you're submitting the right URL, you should open it in a new browser window, to check that your solution loads as expected.

Grading your Solution

When we ‘grade’ your solution, we will do so based upon the following, in order of importance:

1. Correctness of code
2. Clarity of code – is it easy to follow the ‘logic’ of your solution?
3. Succinctness – did you write only as much code as was required? Poorly designed code often contains more logic than is required, to handle special cases that could be avoided with a better approach.

Please note that we are unable to provide feedback on your solution.