Game Developer Test

Introduction

Hello! For this test we would like you to re-create the “Safe Cracker” mini-game from our mFortune game “Buster Safe”. You will have been provided with a reference video, art resources and a boilerplate to complete this. Please see the next page for an explanation of the game’s logic.

Technical Specifications

The game must be written using strongly-typed TypeScript, making full use of OOP principles. The game should be displayed using a HTML5 Canvas. The game scene should be drawn directly on the HTML5 Canvas, and not using DOM elements.

We have provided a boilerplate which can be used as the base of the game.

An Environment Setup Guide has been included as part of this document.

Delivery

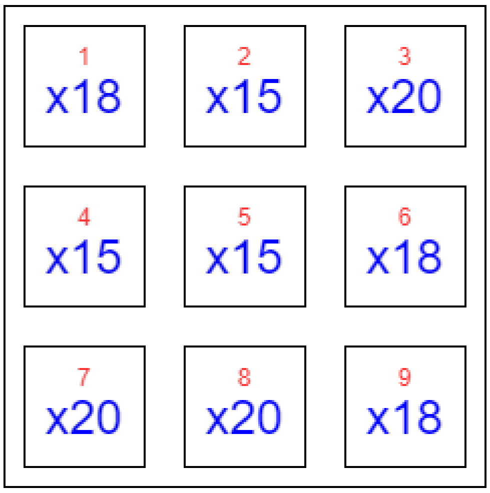
We expect the full source code of the game for review. For your reference, we will be using VSCode to review and run the game.

Good luck! We look forward to seeing what you can produce.

Game Logic

There are 9 safes that can be opened, each one containing a multiplier. The range of possible multipliers are x15, x16, x17, x18, x19 and x20.

In any game, only 3 of the above range will be selected, for example, x15, x18 and x20. Each one of these multipliers will appear 3 times in the Safe Grid, spread randomly like so:



The player presses the spin button, which will begin the turning of the dial. The dial will stop at a random number, which will open its corresponding safe e.g. If the dial stops on number 4, safe 4 will open. When a safe is opened, its multiplier is revealed.

There are up to 4 spins to be completed. The player will continue to press the spin button and open safes until the win condition is met. The game is won when a player matches 2 multipliers e.g.

- Spin 1 opens safe 6, a x18 multiplier is revealed
- Spin 2 opens safe 4, a x15 multiplier is revealed
- Spin 3 opens safe 9, a x18 multiplier is revealed
- The game is complete, the player has won x18 their initial bet amount

Given 4 spins, a win will always occur. This is intentional. This game doesn't have a lose condition. The dial cannot land on the same number twice.

The multiplier is applied to the player's initial bet amount (set in code at initialisation) to provide a final win amount.

Please watch the reference video for an example of this in action.

Environment Setup Guide

Compiler

We recommend using VSCode. It is the compiler we will be using the run and review the game.

Link: https://code.visualstudio.com/

Dependencies

1. Node.js

Node.js is required in order to use the Node Package Manager. Download *either* the LTS or Current version from their website: https://nodejs.org/en/

2. *TypeScript*

In VSCode open a Terminal and type the following command:

```
npm install –g typescript
```

This will install TypeScript onto your machine globally.

Further Reading: https://www.typescriptlang.org/

3. *http-server*

http-server is a node package will allow you to launch and run your game. In VSCode open a Terminal and type the following command:

```
npm install –g http-server
```

This will install http-server onto your machine globally.

Running the Boilerplate

1. Open VSCode.
2. Open the Boilerplate folder.
3. Open a terminal window and type *tsc –w*. This will compile the project and automatically recompile when changes are detected.
4. Open a new terminal window and type *http-server*. This will open a port at which the game can be run.
5. Hit F5 to launch the game.
6. A window will be opened, with the game in view.