Left Field Labs Unity Engineer Take Home

Thank you for your interest in joining Left Field Labs and for taking the time to answer our questions. Principles we value beyond pure programming skills are effective communication, resourcefulness and knowledge. With that in mind please write any notes or assumptions you have made in answering the questions. If there is an area you feel your solution could have been improved given additional time or research add that to your notes.

No one works in a vacuum so use any references (internet, books etc.) you like but please cite any that you use. Be honest with yourself about trying to find a solution you could come up with in a reasonable window of time as you will be faced with more challenging tasks on a daily basis.

In the coding portion feel free to add comments where you feel it's appropriate. Pay attention to formatting and style and treat the code as if it were ready for production, something you would be proud to submit for peer review.

Engineering scenario challenge

Choose **TWO (2)** of the following scenarios and copy them in to a new file. Please fill out answers to the questions and include any relevant thinking or resources.
Submit as a separate document with your name in the title ("ScenarioChallenge_YourName")

Scenario 1:

You are brought on to help with the latest LFL VR masterpiece that has already been in development for 6 months. Your support will help take it from a working alpha to a feature complete beta. You are familiar with the functionality as you have played around with the experience casually and even understand the overall structure from a few offsite lunches with the lead engineer. After being assigned a few "simple" bugs you get to work. Immediately you realize a core piece of the code that can be improved for efficiency is being used throughout the project and has been in the code base for almost 5 months.

- **Questions:**
    - How would you attempt to understand the design of a large code base in a short period of time?
    - What steps will you take to understand the ramifications of the potential refactor?
    - How do you balance the task of fixing the bug with the investigation?

Scenario 2:

You are coding the gameplay of a cool new 3D arena shooter that has a design requirement involving many enemies, bullets and coins to be on screen at once. You notice when initially testing your game the framerate is uneven and the game seems to even freeze at times.

- **Questions:**
    - What steps do you take to deduce the problem?
    - What are some programmatic solutions you could propose to fix the problem?
    - How will you confirm your changes had a positive impact?

Scenario 3:

You're deep in the middle of making what will surely be the greatest ARCore based 3D sticker app the world has ever seen. At the morning stand up you hear that the sound designer has made a pass updating the existing UI sound files while the art team have submitted 10 new 3D sticker prefabs bringing the total to 250! Excited to see the new changes you rush back to your desk and update. After making a new build you notice the size of the .apk has doubled in size, far too big for the changes mentioned!

- **Questions:**
    - How do you go about figuring out the cause of the increase?
    - How will you inform the team of your findings?
    - What steps do you take to reduce the size?

Practical Test: Pumpkin Shooter

The provided Unity project (Pumpkin Shooter) is incomplete. The game is played by using the arrow keys to aim a cannon and spacebar to shoot cannonballs at spooky pumpkins! See how many you can shoot in a limited time, go for the high score!

Your task will be to familiarize yourself with the project and then add missing features that will ensure pumpkin shooter is handed back a better, more complete project than you found it.
The main purpose of the exercise is to determine how you approach C# code structure and data driven system architecture within a Unity project. Complete all tasks listed below and choose **ONE** of the polish tasks to complete as well.
The Unity version used is 2018.3.6f1 which you can download here:
https://unity3d.com/get-unity/download/archive

Submitting your test:

1. Create a private repository in GitHub
2. The first push should be the existing engineering test without modifications.
3. Add 'lfl-engineering-test' as a contributor
4. Feel free to make as many or as few separate commits to the repo while working on the test.
5. Create a README.md file describing:
    a. The changes you have done.
    b. Project structure.
    c. Any other information you may deem useful.

Project Tasks:

1. Data I/O:

    The project needs to become data driven. Create and read data from human readable file or files (JSON, txt, etc) you create. All information you deem relevant that could be changed by a designer or come from a server should live in these file(s).

2. StartMenu:

    As it stands the menu is not very informative or functional. Bring it to life!
    - All fields should be filled in from data read in from a file in the project.
    - The title should be updated
    - The version number should show Version: 0.9
    - Play button should load the game scene.
    - High score should be tracked between sessions.

3. GameScene:

    The game scene is also lacking functionality! The cannon is working but the pumpkins stop spawning after being shot :(
    - Make the pumpkin prefabs spawn at the spawn points already located in the scene. When a pumpkin is shot a new one should spawn.
    - Ensure shooting a pumpkin causes the score to increase and that the high score UI is functioning properly.
    - What pumpkins spawn, the game session length and any other gameplay related variables should come from data.
    - There should not be spawned objects left in the scene past their usefulness.
    - Game over screen buttons should function as expected.

4. Polish tasks **(Choose <u>ONLY ONE</u> of the following to implement)**
    a. Add a feature where pressing the 1, 2 and 3 keys cause the cannon to animate and rotate at a constant rate in order to align with a corresponding spawner. Have any user input disrupt the animation and return control to the player.
    b. Pumpkin shooter could use a theme song! Find a piece of music to play during the menu and have it loop. Make it continue to play un-interrupted while the game scene loads, while the game is running and when the player transitions back to the menu.
    c. The impacts are lacking visual panache. Add a couple appropriate particle effects when an impact occurs in the scene. A cannonball hitting an enemy pumpkin should be one effect. A cannonball or pumpkin hitting the ground should cause another.
    d. There are a few extra cannon prefabs sitting idle in the Prefabs folder. Bring them into the experience in a fun and interesting manner.