2800 Biscayne Blvd
Suite 600
Miami, FL 33137

Software Assessment for Web Developers

Create a browser game mimicking the classic two-player, turn-based artillery game. See https://en.wikipedia.org/wiki/Artillery_game. Each player has a cannon that fires projectiles at the opposing player. The players take turns firing their cannons at each other until one player hits the opposing player's cannon with a projectile, at which point they win and the game is over.

The rules of the game are:

1. The field of battle is 2000 m wide and 1000 m high.
2. The cannons are 1000 m apart, and placed 25% of the width from each edge. They do not move.
3. The cannons are at the same elevation and there are no obstacles on the field of battle. i.e. the ground is flat.
4. When it is his turn, a player specifies the angle at which his cannon fires (0 - 90 degrees) and the initial speed of the projectile, and then presses a “Fire!” button. Use a text box for the initial speed, a slider for the angle, and a button for “Fire!”.
5. Once a player fires, the projectile should be shown flying through the air to its final position on the ground or until it leaves the field of battle. Once it hits the ground, display its distance to the opposing player’s cannon. Show each player’s projectile movement to both players.
6. Projectile movement should match the real physical movement given the parameters of the field and equipment. If it would take the projectile 5 seconds to reach the target, it should take 5 seconds.
7. If the projectile falls within 3 m of the opposing player’s cannon, it is a hit and the firing player wins. Display “You win!” or “You lose!” to the appropriate player and end the game.
8. If the projectile falls farther than 3 m from the opposing player’s cannon or flies off the field, their turn is over and they must wait until the other player fires before getting another turn.
9. The projectile is unpowered while in flight. The only forces operating on it are drag and gravity.
10. While the projectile is in flight, gravity operates on it. Assume gravity is 10 m/s^2.
11. The projectile also experiences drag and wind resistance. Assume the projectile is a 10 cm diameter sphere, weighing 1 kg, and the air density is 1 kg/m^3.
12. Also assume that there is a wind blowing parallel to the ground of 5 - 30 m/s, chosen randomly by the server. Display the wind direction and magnitude to the players.
13. A player may possibly hit his own cannon by firing at too great an angle, in which case he loses.
14. Use whatever graphics you like for the cannons, projectiles, etc. They can be very simple.

Create a web service that serves the game board that represents the field of battle and operates the game as described above. What parts of the functionality you do on the server and what parts you do in the browser are up to you. However, please document your reasons for choosing the architecture that you use.

2800 Biscayne Blvd
Suite 600
Miami, FL 33137

By pointing a browser to the server IP address and port (you can choose the port), the server should detect the request, create an instance of the game if necessary, and connect that player to the game. If a player from a different browser connects, they would become the second player in the game. If a third browser connects, a new, separate game will be created for them, which they could play against a player in a fourth browser. Similarly, a fifth and sixth player could play in a third game. The web service should maintain each game instance separately. If a player loses connection and then regains it, they should be connected to the same game as previously, and the game should be in the current state for them. If a player's game has ended, and they refresh or reconnect, they should be attached to a new game.

General Guidelines for both parts:

Your solution should be easily executable, and well documented. Document all algorithms in full. Provide instructions to install and start your web service.

Ways to Improve your Chances:

- Good code design & structure
- An object-oriented solution
- Good documentation!

Requirements for submission

- Turn your result in by the end of the second day. Earlier if possible.
- Please use any standard development language (e.g. C/C++, Java, Python, JavaScript or equivalent). LabView, MATLAB or equivalent are not considered to be standard development languages.
- Your web service must be able to be installed and work on a Windows machine.
- Attach all documentation and source code in an email to **jtieman@neocis.com**.