# Having a shared score board

(A student worksheet for “Flappy Glitch)

Ok, so you now have your own working version of “Flappy Glitch” hosted on PlayCanv.as.

You can run the game in your browser, and it “saves” your score. But if you run your game on your phone (or another computer), is your best score still there?

How can you “reset” your high score back to 0 when you run the game?

What happens in the game when you try to “share” your score?

Where does the code for “Flappy Glitch” come from?

*Where* is the code for “Flappy Glitch” actually running? On which computer?

Does it work without a network?

## Clients

If you’ve done any computing, you should have realised that Flappy Glitch is essentially **client side code**. Your computer requests your game from the PlayCanv.as server. The PlayCanv.as server sends a pile of JavaScript, and your browser runs it. Other than the initial request for code, your computer never talks back to the PlayCanv.as server while you play the game. The high score persists on your computer because it is using browser storage. You can see that high score using the browser development tools.

**TASK 1:**

***Find where your high score is actually stored. What is the name of the score?***

*(Hint: View->Developer Tools in Chrome, look for application)*

## Servers

In order for things to persist online, they need to be stored in a server. A server is just a computer, but it is programmed to respond to specific requests. The PlayCanv.as server for your game responds to requests for games, but it doesn’t store or report high scores. We’ve set up such a server on the RGU campus network. It lives at <http://soc-flappy.rgu.ac.uk:8080> and you can check out its super-default Flappy Glitch if you like just now.

We’re going to use a different server to share the flappy high scores. But first there are a few checks and changes for your PlayCanv.as game.

## Task 2: Changes to make to your flappy glitch

You should have seen earlier that your score is currently stored in domain-specific local storage. We need a way to get it out of domain-specific local storage. The following function makes the PlayCanv.as code signal the last score to the browser window.

Somewhere in game.js, declare a function like this:

function sendScoreToWindow(score) {

window.top.postMessage(score, '\*')

}

Then on game:share (where it currently puts up a Twitter popup), change it to call the function like this:

sendScoreToWindow(this.score)

Pam’s project with the code changes to game.js only is here:

<https://playcanvas.com/editor/scene/1847802>

I commented out all the Twitter code, but no reason not to delete it altogether, or even leave it in place where it can annoy you.

Really, it shouldn’t matter where you put that code, but putting it where the Twitter popup previously went means that you’ll actually have to click “Share” within the game before you can upload your score. All the control, none of the convenience. There might be a better place to call the sendScoreToWindow() function, but you can always change it later.

In any case, publish your game and check that everything still works as intended.

Note that when you publish your game, PlayCanv.as gives an option to run the game in an iframe and there are a couple of lines of code. You will want these for later.

## Your own front end

We’re going to write some html and JavaScript now to construct a page to play the game and scoreboard. If you’ve written HTML and JavaScript before, this should be straightforward. If you haven’t, there is a starter solution here: <https://github.com/PamJohnston/flappyBackend>

The GitHub (<https://github.com/PamJohnston/flappyBackend>) contains a few things. It is a complete client and server and it is currently configured to run front end and back end on a single machine. You can see the code for the backend if you’re interested but we might not look at that today. Second, you can find an html and css file in the “frontend” folder. This is your front end.

First thing to do, launch the html in a browser, see what it does. That’s not your game.

**Task 3: Make it your game.**

(Hint: the iframe code you saw earlier on PlayCanv.as is in here somewhere, you just have to change that link).

**Task 4: Connect to our shared scoreboard**

In the first instance, we’ll all use the same backend. In the GitHub code, the backend’s domain used is <http://127.0.0.1:8000>. That assumes that you’ll be running your own backend on your own machine but you’re not. Change all of those to the address of our server (above).

**Task 5: Check it all works**

If you refresh your page, you should see your game. You should also see a scoreboard with some scores. Pam is really bad at Flappy Glitch. And you should be able to play your game, share your score and upload your score to the scoreboard. What are the limitations?

**Task 6: Too many scores?**

You should be able to edit the JavaScript to make it only 10 scores long (or only 5 scores long). There is a “for” loop that scans and displays all the scores that the server sends, it doesn’t have to. Make it shorter. You can maybe also filter out some scores using your own criteria.

**Task 7: Tidy up the console logs (optional)**

Open up the developer tools and look at the JavaScript console. Everything that uses console.log() gets printed here. Developers keep lots of secrets here. Tidy it up or use it to explore the code.

**Task 8: Make it your page (optional)**

It’s a pastel pink page and it looks pretty dreadful. You can change it using the CSS and html code if you want. <https://www.w3schools.com/css/> will help you to know which options are available. Stick to basic CSS and HTML for now.

**Task 9: An easier UI**

Store your player name in local storage and have the game send the score directly to the scoreboard whenever you hit “share” on your game. Hint: localStorage.setItem() is useful as is the code in the uploadScore() function.

**Task 10: What is “comment” about?**

The HTML/JavaScript code contains something about “comments” – they can be received but they aren’t posted. See if you can figure out how to post a comment to the server.

**Task 10: Spinning up your own server (definitely optional)?**

You’ll need to change all the urls from the RGU server back to local host for this to work properly. In PowerShell, spin up a database using

*mongod --dbpath .*

And then in a new PowerShell (with the database still running and the folder pointing to the backend) try:  
*npm install*

*node index.js*

Visit the address that it says to see your own backend running on your own computer. This will persist high scores and will also be accessible from other computers if you can find out the name of your computer.