ECE 371
Spring 2019
Final Project Report

Flappy Bird in SystemVerilog

Names: Sidd Jadav, Hannah Ho

Introduction:

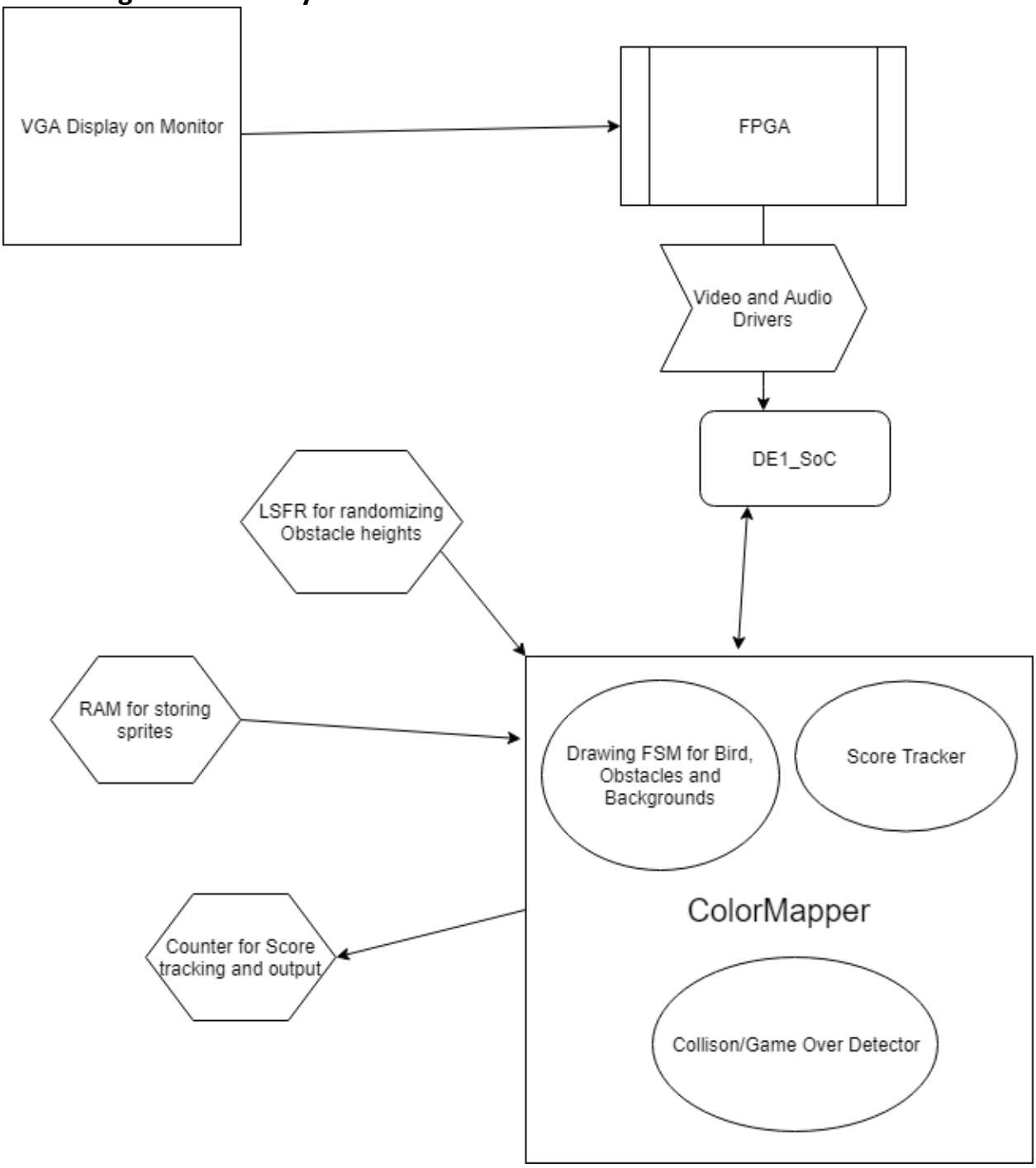
This was the final project for ECE 371 in which we were tasked with creating a project that involved using a keyboard/mouse or a camera. We also had to incorporate a VGA display and sound output that built on our lab 5.

Me and Hannah decided to go with a side-scroller game since we felt like it would be the most fun to work on, and cool to show off to our friends. We researched multiple games and decided on flappy bird because it felt like the most accessible game. Regardless of whether you're a hardcore gamer or casual, flappy bird is known and enjoyed by all.

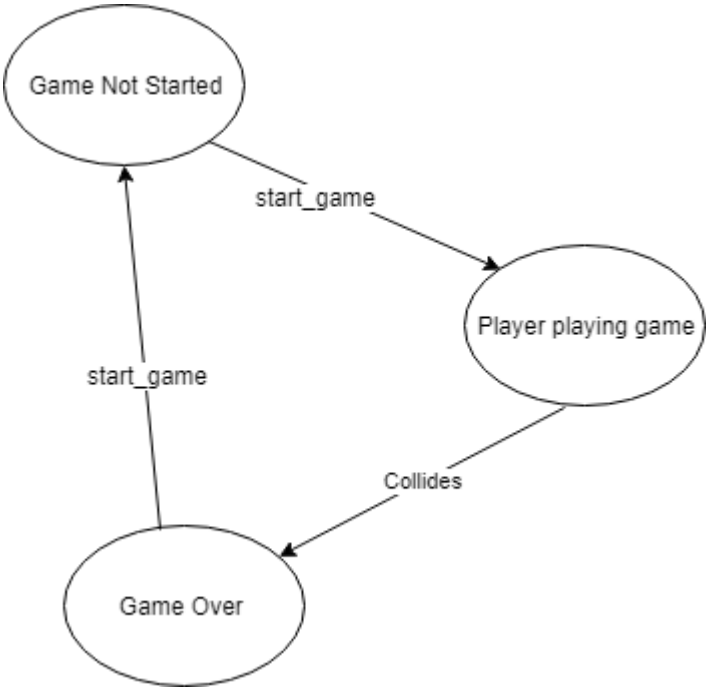
Now onto the actual specs of the project. In this project, we use the mouse as an input for the bird jumping, the VGA display for the game, and the speaker for sounds for when the bird jumps up. I want to make note here on as to why this is different from a ECE 271 project. Our game has an intro title, a game over title, and the actual bird drawn for the game (as approved by the TAs). It is not just a simple block dodging obstacle game. You will see as you read the rest of the report.

Procedure:

Block diagram for the System



Drawing FSM



Major Components of the System:

1) DE1_SoC
2) Color mapper
3) Ram
4) LSFR
5) Counter
6) Driver Files for mouse, speakers and VGA display

DE1_SoC is the main driver for the system that connects the various other systems together. It holds together the code architecture, connecting the Audio and Video Drivers that provide us the outputs on the VGA display and speakers. It also connects to the mouse through the ps2 drivers for mouse input.

The Color Mapper is where EVERYTHING basically takes place. I will split its functionality into 3 parts: The Drawing FSM, the collision detector, and the Score Tracker. I will explain how all 3 work separately and how they come together to run the entire game

The Drawing FSM has 3 states. In the beginning the state is S1(Game not started). It shows the bird, the game title and our names. It continues displaying this until the user presses the start button (assigned to KEY[1]). Once pressed the FSM moves to S2(Player playing game) state. In this, the bird is now controlled by the mouse, whenever the left mouse button is pressed, the bird moves up and when released, the bird drops down. There are 2 obstacles which have a constant space between them, and the upper and lower heights for both are randomized using 2 LSFRs. The 2 obstacles spawn at a position on the right and move towards the bird to give the effect of motion, and the user must maneuver the bird without hitting the obstacles in order to score points. If the bird collides, the FSM moves to S3(game over) and a pop-up GAME OVER shows up. In order to restart, the player must hit KEY[2] again. Personally, I do wish I had split up the different aspects of color mapper into different modules but given the time constraint and during prototyping, I did not think of it as a top priority.

The Ram module is where all the sprites were stored. So this is the interesting part. I used a python script I found online(which I will attach with the code) that takes in an .png file and then reads each pixel of the png(from left to right, then top to bottom) into a txt file in the format RGB where each color was designated with 8 bits. Now, having chosen the right images, using the script to convert them into text files, I created a ram logic of the right size (depending on the image size) and read it onto a logic RAM using $readmemh. I then used the formula below to figure out where to draw each sprite. For example, in the case of the bird, it was:

```
//draw bird at 150,150
if ( (x >= 150 && x <= 150 + bird_w) &&
   (y >= 150 && y <= 150 + bird_h) ) begin
      red <= birdROM[(x - 150) + (bird_w * (y - 150))][23:16];
      green <= birdROM[(x - 150) + (bird_w * (y - 150))][15:8];
      blue <= birdROM[(x - 150) + (bird_w * (y - 150))][7:0];
      birdOn <= 1;
end
```

The birdROM read the text file and was passed to the color mapper. Using this similar functionality, I drew the ground, the sky, the Title and Game over Texts, and the obstacles.

The LSFR had an enable signal that when activated, would produce a new random variable that was used to create random heights for the obstacles.

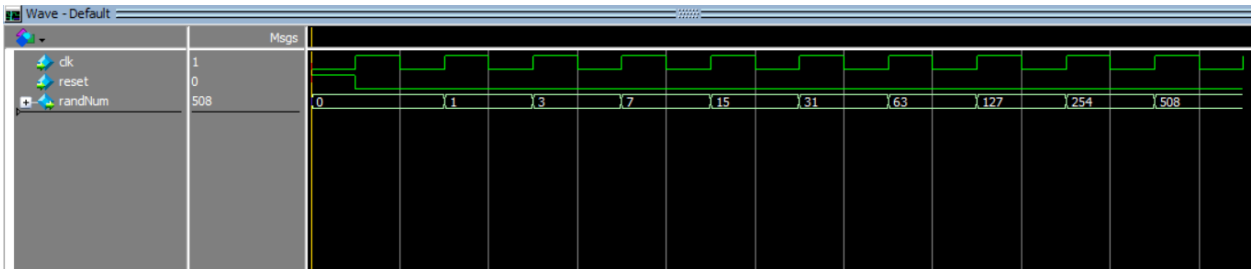
Every time the bird passes the obstacle, the counter module received a signal from the color mapper asking it to increment the score, which was then displayed on HEX0-HEX1.

The driver modules were used as provided to us and no changes were made.

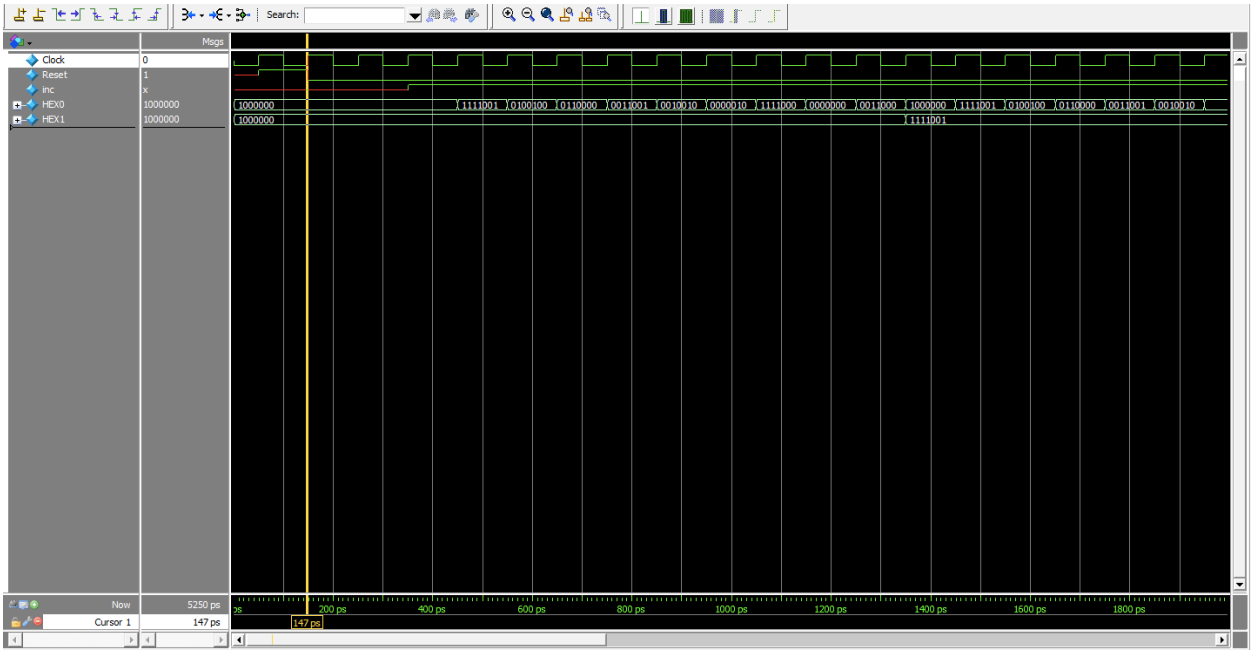
Results and Simulation:

We only used modelsim for the LSFR, and Counter modules due to issues with modelsim running with the video/audio drivers for DE1_SoC and for color mapper, due to us using 2 different clocks, modelSIM would not work. I talked to multiple TAs and since neither could figure it out, I moved on. I eventually managed to make the whole thing work so simulation was not needed for those parts.

For the LSFR, I used a 10 bit LSFR and it produced random outputs as shown below. The simulation is attached below.

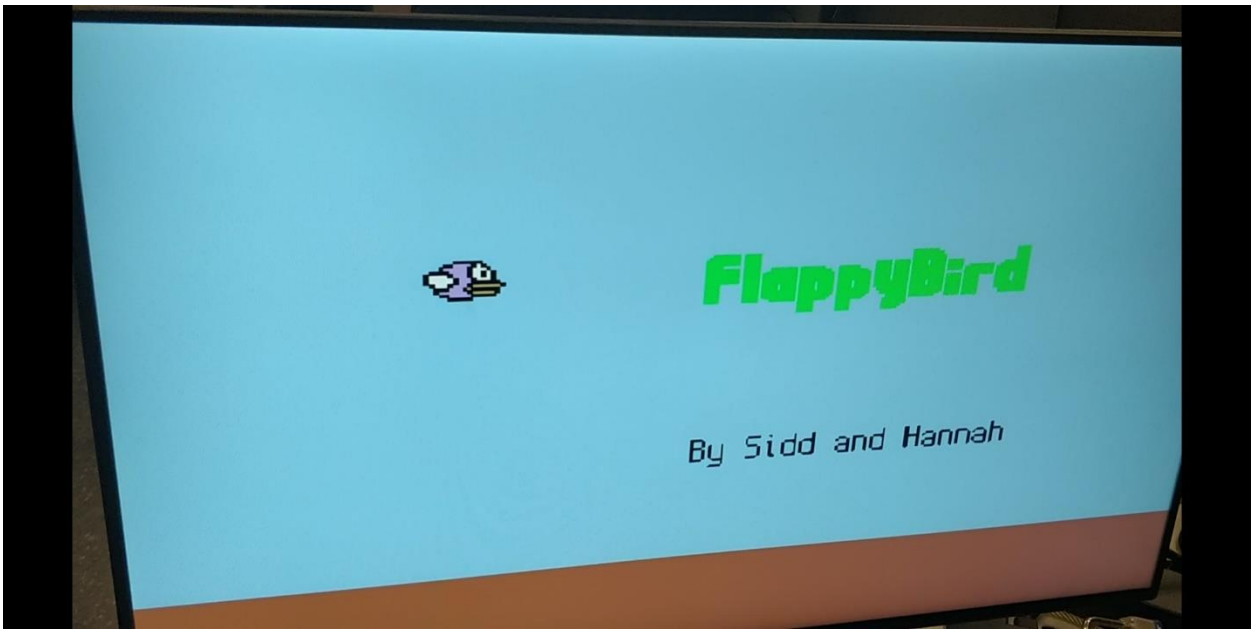


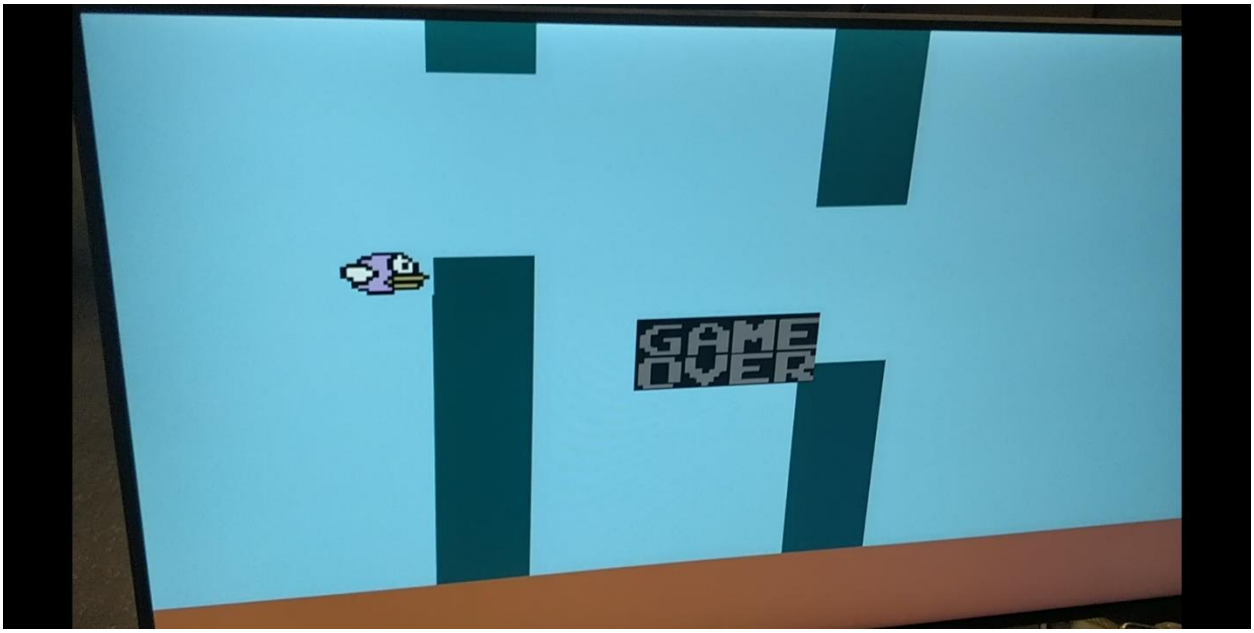
The counter simulation is attached below

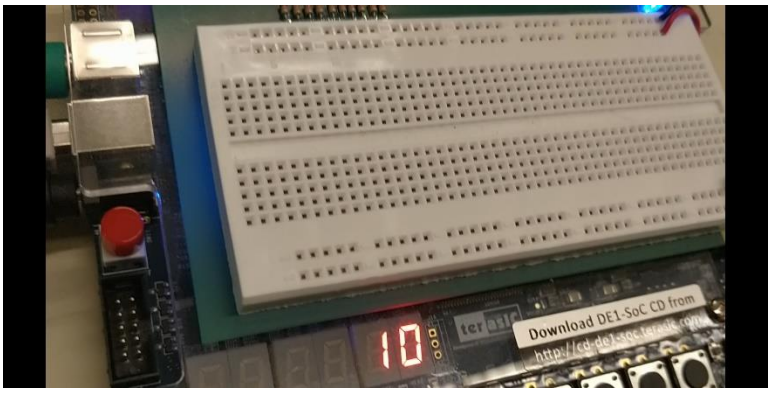


To figure out control and game mechanics, I played around with the various positions of the obstacles and the bird position, the force required to move the bird up/down and finally decided on (-2) for the x-movement of obstacles and (+ or -1) for the y-movement of the bird when going up/down. It just felt right in terms of the controls and felt most natural. I did not want the game to be easy, so the obstacles were paced faster. We tested the score counter by simply running the game and seeing the counter increment after every obstacle crossed.

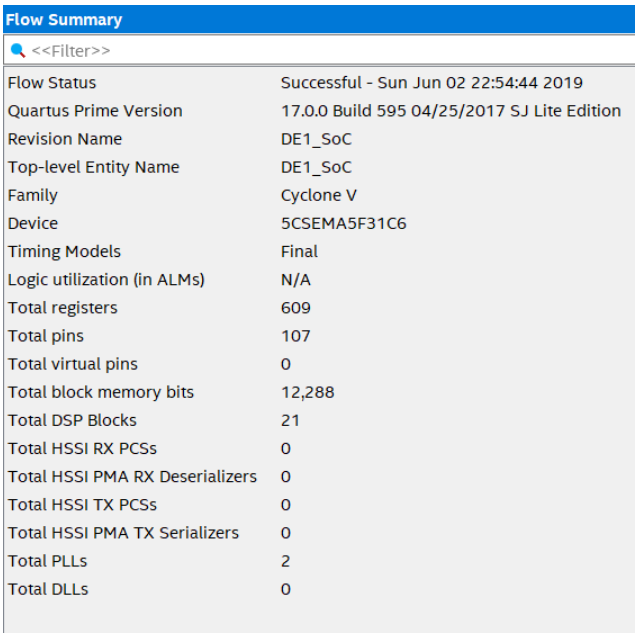
I have attached results below for the different screens/states of the games


FlappyBird
By Sidd and Hannah


GAME
OVER


10
Download DE1-SoC CD from

The flow summary for the whole project is below:



Flow Summary

<<Filter>>

Flow Status	Successful - Sun Jun 02 22:54:44 2019
Quartus Prime Version	17.0.0 Build 595 04/25/2017 SJ Lite Edition
Revision Name	DE1_SoC
Top-level Entity Name	DE1_SoC
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	N/A
Total registers	609
Total pins	107
Total virtual pins	0
Total block memory bits	12,288
Total DSP Blocks	21
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	2
Total DLLs	0

Conclusion:

Overall it was a super fun project to work on and will be super useful to me in working on other games/FPGAs in the future. I had such a great time with everyone this quarter. I have never had such a supportive group of peers before and it was amazing to work in the labs. The TAs were also definitely super helpful. Shout out to **Ling** for being so much fun and informative to interact with. His no-bullshit answers were hilarious. Working on the lab, I realized how important it is to learn from everyone. And ofc, thanks to the graders <3. I've never felt like my points were being unjustly cut and always got the points I deserved.

Problems Faced and Feedback:

When we first started I could barely get a purple screen to show up. The issue was that I was using the old VGA buffer and it was significantly harder to work with. It was only a couple of days later that I saw the new driver files and after that, the project flowed easily. Another great thing I picked up from the internet was tools to convert a png file into a text file containing info on each pixel, all automated with python. It was exhilarating and saved me a TON of time. There was no way I could have done the bird drawing otherwise. It was an amazing trick and I specifically would recommend you guys check this link out for the next batch of 371: https://github.com/Atrifex/ECE385-HelperTools. Overall I spent around 50-60 hours on

the lab but every second was worth it. That's all the feedback I have for you guys : build more tools like the one I linked, it enables us to make much more creative projects and impressive ones too, it's a win-win for both the instructors and the students. Once again, thanks for the great quarter. Hope you have a great summer ahead