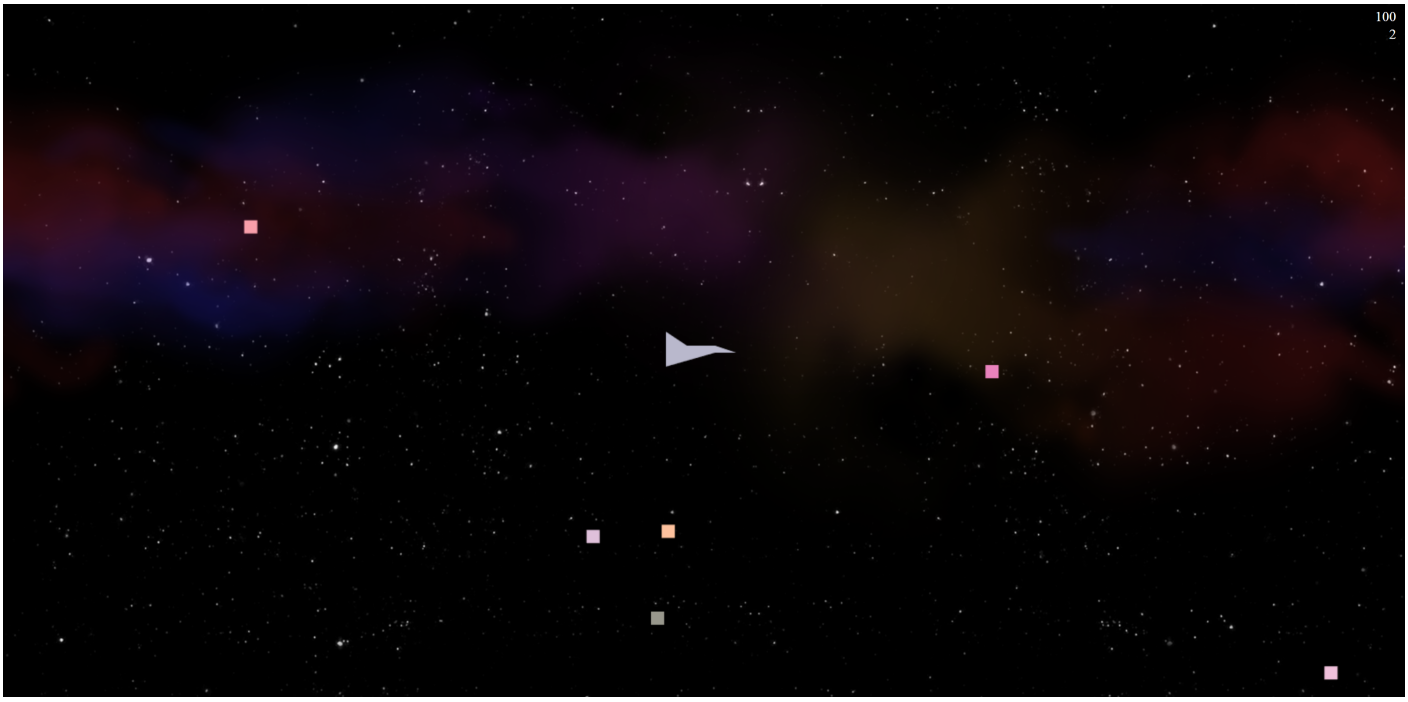
CS460 Fall 2019
Name: Loraine Franke
Student ID: 01881004
Due Date: 10/02/2019

Assignment 4: A Vanilla WebGL Game!



Part 1 Coding: Extend the `createAirplane` method. (25 points)

The airplane will be created out of 4 triangles.

Part 2 Coding: Extend the `createObstacle` method. (25 points)

Part 3 Explaining: Detect collisions using the `calculateBoundingBox` and `detectCollision` method. (20 points)

`calculateBoundingBox`:
Six variables with minimum and maximum value are created for each dimension x, y and z. These values will define the borders of the bounding box. The current variables are calculated by adding up the vertices and the offsets (since the airplane is moving). With the for-loop we loop through all existing vertices of the airplane and compare them each time with the highest value (maximum) or smallest value (minimum) that has been found in the corresponding dimension. The maximum and minimum vertex value will be updated until we looped through all vertices. Then we get returned minx, maxx, miny, maxy, minz, maxz which will create the box. The bounding box of the airplane is later calculated in the animation function with the current_offset and the current_vertices and is called bbox_plane.

`detectCollision`
Collision detection is performed by either setting the variable $collision$ to false or true. If the collision is false, the game

will continue. The collision variable is set to true by the following conditions for all three dimensions: if the current point in dimension x is greater than the minimum x of the bounding box and smaller than the maximum x of the bounding box (means that the point will collide with the box). Same for y: if the point is greater than miny and smaller than maxy, it will collide with our bounding box. The same if condition is set up for the z-dimension. Every time a point of the obstacle is in the bounding box, a collision is detected.

Part 4 Coding: Extend the `window.onkeyup` callback. (20 point)

Part 5 Cleanup: Replace the screenshot above, activate Github pages, edit the URL below, and add this PDF to your repo. Then, send a pull request for assignment submission (or do the bonus first). (10 points)

Link to the assignment: `https://lorifranke.github.io/cs460student/04/`

Bonus (33 points):

Part 1 (11 points): Please add code to move the obstacles! Flying the airplane around static obstacles is half the fun. The obstacles should really move! Please write code to move the obstacles every frame. The obstacles should just move in `x` direction from right to left to create a flying illusion for the airplane. This can be done with little code by modifying the offsets accordingly at the right place!

Part 2 (11 points): Make the obstacles move faster the longer the game is played! Right now, the game is not very hard and a skilled pilot can play it for a very long time. Currently, the scoreboard updates roughly every 5 seconds. What if we also increase the speed of the obstacles every 5 seconds? Please write code to do so. This can be done in one line-of-code!

Part 3 (11 points): Save resources with an indexed geometry! As discussed in class, an indexed geometry saves redundancy and reduces memory consumption. Please write code to introduce a `gl.ELEMENT_ARRAY_BUFFER` for the airplane. Of course, we do not need to change anything for the obstacles since a single vertex does not need an index :).

Sure, but it is even more efficient to use gl.TRIANGLE_STRIP instead of an indexed geometry.