

**COMP 410 – Term Project Report**

**INTERACTIVE AQUARIUM**

**Tümay Özdemir & Necati Barış Altınbaş**

# Motivation

In this project, our main purpose was to implement an interactive aquarium that has most of the main components of an aquarium with the addition of a feeding interaction and other natural sea-life movements. Our main motivation was providing a beautiful 3D graphics program to which we canalized our knowledge that we had gained throughout this semester. In addition to this, the movie named “Finding Nemo” had mostly been our deepest influence for this very project, in that we intended to create a program simulating a simple version of the movie so that it could be used as a starting point for future animations with a slight modification utilizing concrete texture mappings and light shadings.

# Project Design

In this project, we had the intention to visualize an aquarium that putatively responds to some specific user interactions. We had built up our aquarium such that it would contain various types of sea creatures along with its sea habitat. Besides, we got our fishes to be inclined to be fed by their visitor, namely the user. We have used the shader-based OpenGL as well as C++ as the programming language for our project. The fundamental problem in our project had come out to be implementing the movements of sea creatures in a random fashion that would be observed as a series of smooth animation without compromising the reality aspect of the natural aquarium realm. In as much as it was hardly possible to be able to find a texture element that would be a perfect fit for any of the sea beings in the aquarium, the significant challenge that it had brought along easily outweighed our desire and intention to implement such shading or textures into our project.

We started off with a search of web for some nice-looking fish models which we thought would be relevant to and useful for our purposes, this was when we had the realization that it would be quite difficult for us to find both free and useful models until encountering the website called free3D.com, from which we downloaded a small fish, octopus, urchin and seashell in the .obj format. Then, in order to be able to read the models, we resorted to MeshLab so as to convert the models into the .off format. After all this, only seashell had a texture and all the others had no texture bindings due to the fact that all the sea creatures were comprised of free models.

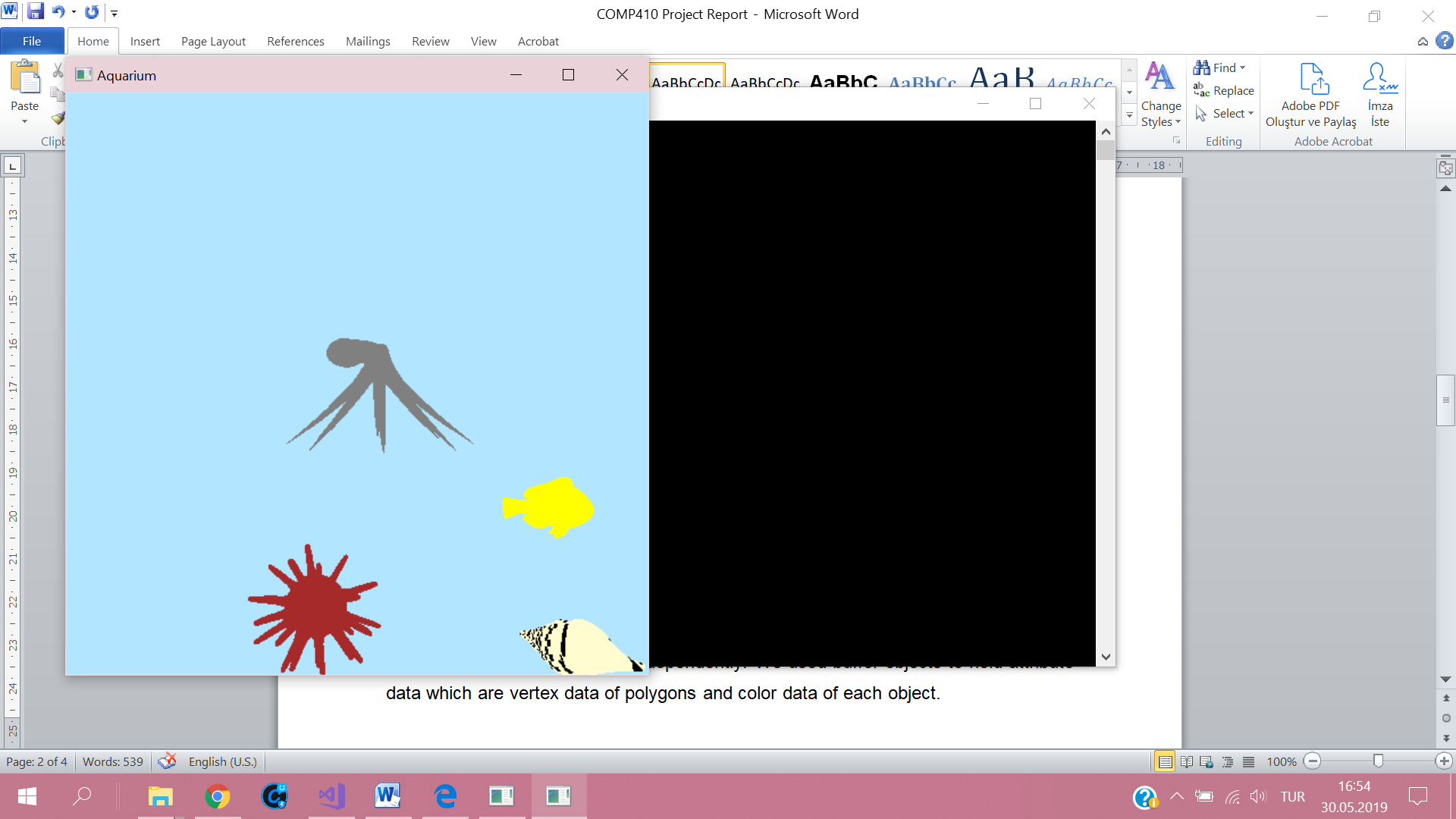
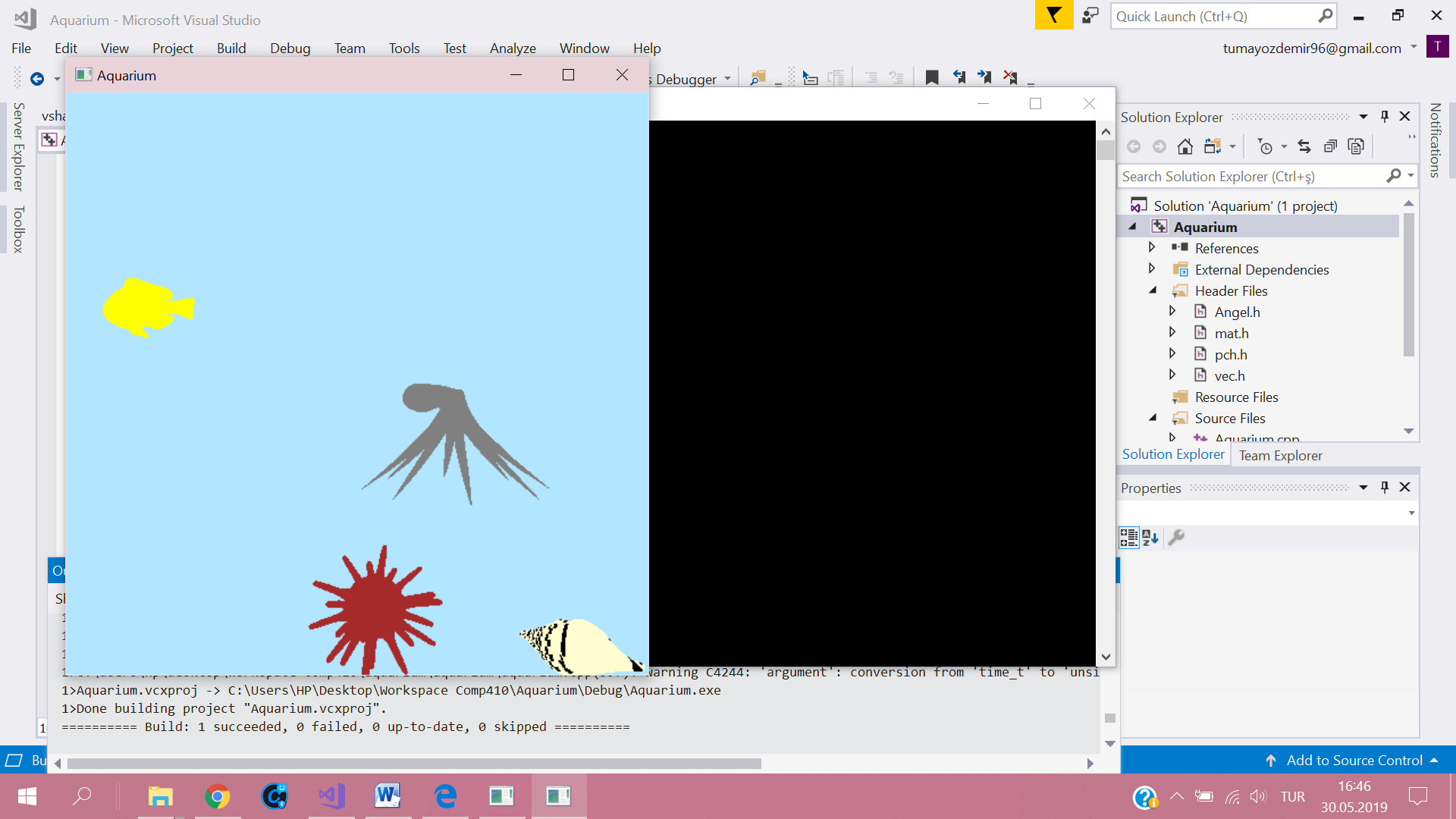
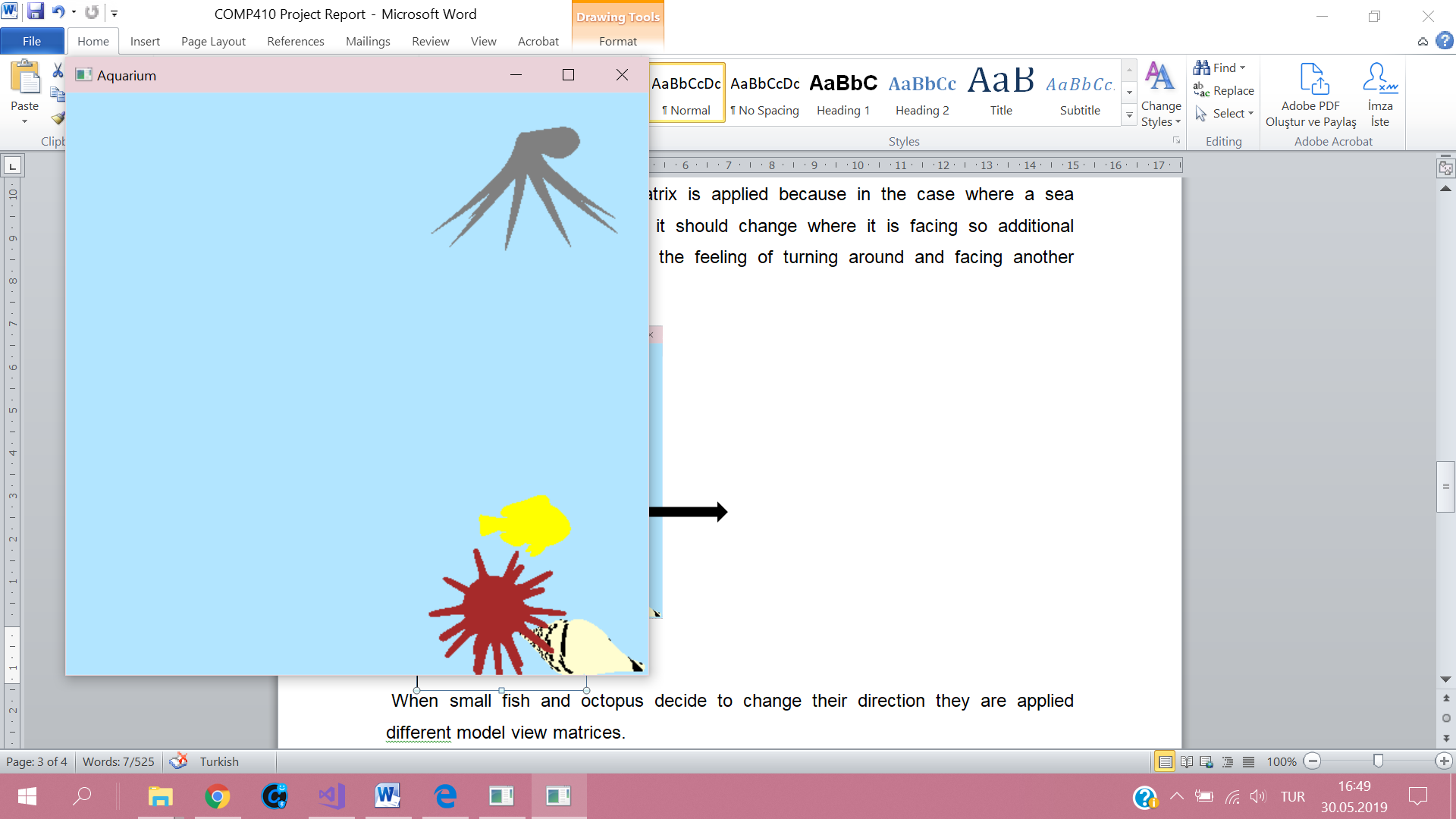


Figure 1: Sea creatures have different colors and seashell has texture

After reading models, we created a separate buffer, VAO and model view matrix for each sea creature in order to be able to handle the movement of each object independently where buffer objects were used to hold the attributes consisting of both vertex data of polygons and color data of each object, apart from having unique callback functions to handle the idle, reshape, display and mouse events.

In the implementation, vital calculations take place in the idle function. We created separate global displacement vectors for each sea creature which is randomly decided in the idle function. In the idle function, we find a random movement with a random direction. For that, we used counters for each object which decides on the direction of the object. There are counters\_x[] and counters\_y[] arrays. In these arrays direction information in x and y directions are hold for each object. According to the counter value, different model view matrix is applied as in a case where a sea creature changes its direction, it should change the direction it is facing thereby requiring a need for an additional rotation to be applied to it which eventually invokes an impression of turning around by means of facing another direction.

Here octopus is facing to the left.

Here octopus is facing to the right.

Urchin has a different movement than the octopus and small fish. We provided a concatenation of rotation and translation to create an impression that it actually walks on its spikes on the ground. Also, we handled the sized of the aquarium so that no creature can leave the aquarium. We assumed that the aquarium is the window so no creature can leave the limits of the window. Whenever they reach to the limits in x or y direction, they turn around and continue their movements inside the window.

We used perspective projection to arise a 3D view. The effects of perspective projection are easily recognized in the urchin’s movement and on the octopus’ legs.

We used mouse callback function to handle mouse events. When mouse is pressed and hold fish and octopus come to the cursor location as if we are feeding them and they are moving to the bait. We obtained cursor’s pixel coordinate in the mouse callback function and we ran an appropriate computation separately on x and y components to find the location in the view volume of perspective projection.

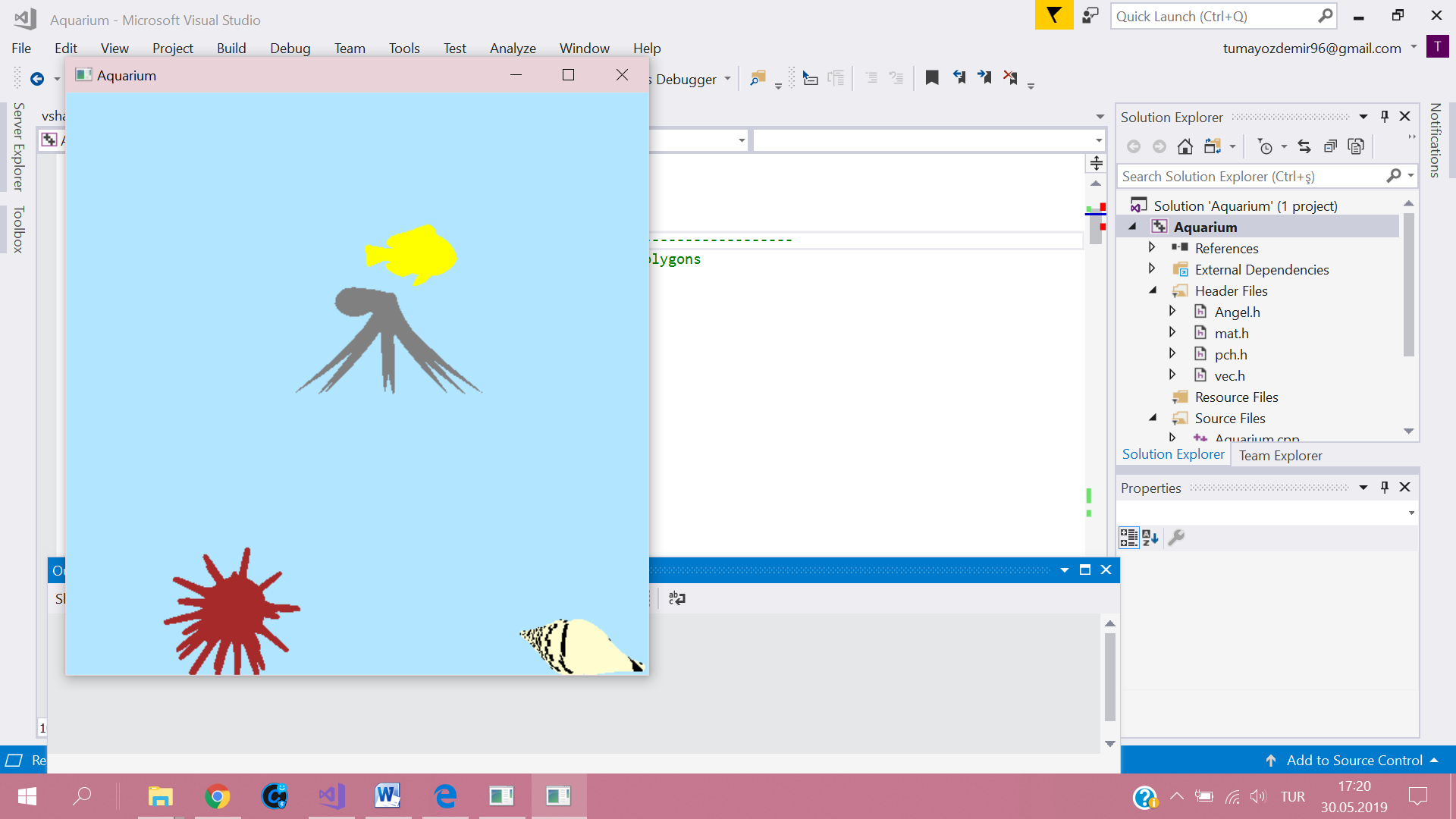


Figure 2: Fish and octopus are moving to cursor (cursor was on the right top)

Finally, we used reshape function to handle changes of the size of the window. Whenever the window is resized, sizes of creatures change, this time limit of the aquarium again becomes the window size so that no creature can leave out of the window and they continue their actions inside the window.

# Conclusion

We managed not only achieve a smooth animation of the aquarium with its habitat but also to implement a lifelike interaction through mouse. When the mouse left button is clicked and hold, small fish and octopus move to the cursor as if someone throws some fish bait at them. For future use, it can be improved via an addition of texture mapping as well as light shading on objects. (PS: We fail to provide any texture on some objects due to their lack in the most of the free models used in our implementation.)