Summary of the Game

The game turned out almost the way I imagined it from the beginning. You are a small pirate ship made of paper; you shoot pebbles at peaceful merchant ships (but they can defend themselves since you go under a pirate flag). The camera is controlled using the mouse and the two canons on the port and starboard sides are fired with the two buttons on the mouse. The boat itself if controlled with the WASD-keys and P for pause.

You only have 300 seconds to score as much loot as possible, your life can be refilled with scattered paper around the sea. The sea was meant from the beginning to be infinite but as I started to do the actual development it turned out to be a tough challenge and, with a sea with limits, you can do more meaningful designs. And that is way more fun.

2. Design and implementation

Architecture you used for the game specific code

In this game, a rather small one, I used a modular architecture and somewhat Ad-hoc. The reason for this is that the game is small, and I was the only one working with it. I added scripts to certain GameObjects that had a relation with that particular object such as the Ai-script for the enemy boats, EnemyCollition for the enemy boats, PlayerController on the PlayerBoat, PlayerCollition on the Player Boat and so on. I really tried to use short methods and to use as little dependency as possible but I can only say that it is hard when you struggle with the scripting and Unitys interface at the same time.

However, I can understand why this isn´t an optimal way of doing things when the game is big and complex (especially if I´m not the only one working with it). Then I would probably have chosen the DAG architecture. To accomplish the Layered architecture in Unity wasn´t all that clear in the lecture and frankly a bit hard to understand without having done anything in Unity whatsoever. A better approach in these lectures would be to refer them to the Unity Engine, that way things would be much easier to understand. Hence I had to apply the general knowledge I had from coding in general and use it at the same time I tried to learn Unity at the same time.

Design patterns and data structures

I used Lists to keep track of the highscore-board and I used the regular Update() for the rest. The reason for this is that at any given time, there is only at max somewhere around 15-20 Updates that has to be addressed and there were no Events in the game that affected a lot of GameObjects at once. When an enemy ship sank, it hit the sea floor and that triggered a random spawn point above sea level to ether spawn a small ship or a big ship. Nothing else in the game needed to know this except the KillZone and the SpawnPoint.
The only memory consideration I made was to destroy the cannonballs and sunken ships. I used the Unity Destroy()-method to accomplish this and also, when a round is completed the game simply reload the whole scene I had the garbage collector take care of the scraps.

Collisions and geometry

Collisions were used both for the boat to float, for rocks, sea, enemy ships and the sea floor (the KillZone). I used a mesh for almost everything since my models were low-polygon and it wouldn´t affect the speed of the game very much. However, when possible I used the box-collision whenever possible to save on calculations as mentioned in the lecture. It also depended on type of collision if I

used the OnEnter. For example, you can ram the enemy boat and cause -3 damage to the enemy ship but only -1 on your boat. To avoid continuous enemy damage and player damage I only used the OnEnter.

To make the Ai-system I used rays (the Unity-build Ai system was way too simple for this). Both to make the enemy boats "see" and avoid collision but also so auto-fire the canons. If the eye-ray were to spot an imminent collision, it would execute the Avoid()-method inside the Ai-script. The canons also used rays. If the Player were to be hit by a ray, the canons fired automatically. The Ai-script and the canon-script are totally unaware of each others actions. If I were to make the Ai more sophisticated this could have triggered a series of events but since they are just peaceful merchant ships they weren´t supposed to engage in battle unnecessary.

Textures, shaders, materials and lighting

I used the textures and shaders that came with the free assets models that I used. There had to be some modifications thou. For the sea I added tree special shaders. One to make the sea render in a certain queue to avoid the boats to have water in them. The other was to make the boats float. The floating is a borrowed script that used the Archimedes principle and was way over my math skills to make on my own. But in a sense, it just used a special "float"-mesh to calculate the buoyancy of my boats using a up-force on their rigidbodies. The third script is a borrowed one and that is to make the low-poly water have those nice bloom-effects when the light shine straight at them. To light the scene I used two different directional lightning (one for primary and another for fill) and a light for when the canons were firing (together with a small particle systems that I downloaded and adapted somewhat). It seemed enough for this kind of game.

Animations

No special animations were used in this game.

3. Discussion and reflection

If your resulting project is different from your approved game description, describe why and how, and explain how those changes affected the code design.

The only differences I had was the infinite sea, because of the reasons mentions before, I only used one type of enemy ships but with two sizes, I added (per request) a highscore board, a time limit, a settings dialogue, a pause-function and a "main menu". I also used just once Scene to avoid loading times but that was unnecessary since the game loaded rather quick anyways. Instead I had the main menu to be a separate part of the set and when the game began, I simply deactivated the menu part and activated the game view.

These above descriptions affected my previous idea and how I would conde it in many ways but as I said, some of my code base were somewhat unorganized to begin with before I got the hang of Unitys script based system, how GameObjects acted, what a bloody Prefab was and how to use it, how to edit models outside of Unity (I used Blender to make appropriate changes to the mesh-models when needed), Coroutines, the Inspector et.c.

How you solved any other implementation issues that you had.

Most of my scripts are mine and the few I downloaded were ether in their separate asset-folder OR if they are in the script directory, I've added a comment where I got them from.

The biggest issues I had was the water, to make the boats float, sink and not render water inside them. This was way over my head (no pun intended) so here I had to use a script I got from the asset store. The rest I simply got inspiration from the Unity Forum, I asked a total of three questions and even answered one myself for another developer as I learned more.

The camera script was also borrowed but I had to make a crucial adjustment to the zoom. The code from the beginning caped the zoom by adjusting it if the camera came to close. That meant that if I got too close, the camera would go back to the limit position again and again making the zoom choppy. I implemented a new bool-method that would stop the zoom all-together instead making it work as intended.

What would you change in your design and implementation?

I would use another architecture to begin with and I am already sketching for another small game but a much simpler one. The only object with this is to learn to use a better structure when building. The main reason, as mentioned before, is that now when I know how Unity works it is much easier to apply this knowledge from the lectures. I did have programming knowledge prior to this course but that only helped with the syntax and libraries I could use. The Unity-part was somewhat a new "language" to learn.

Side note: All sound effects and the music were downloaded as "free to use" without limitations from Youtube (bg-music ony) and freesound.org.