Build doesn’t work, unity fails to export all required assets.

**Object Pooling**

Started off with what’s basically a Struct, called Pool, holds tag/prefab/poolSize information. ObjectPoolController has a list of the Pool “structs” and a dictionary of runtime pools. Had the ObjectPoolController make itself a singleton so any script could reference it very easily. The controller has functions for Initing/Removing pools, and “spawning” objects from the pools. The driver just does randomly spawns/deletes objects for demonstration.

This optimizes memory management. Firstly on object init there will be no frametime spike from memory allocation. Secondly there is no memory deletion from the object being “removed” from the scene. This helps with reducing memory fragmentations and less garbage collection causing frametime spikes.

**Command Design Pattern**

Not implemented, no time. Started off with a basic interface which holds the base command Execute function. Then made a InvertMovement command which depends on the interface. When two ducks are missed, the InvertMovement command will be added to the command buffer in the CommandInvoker. When the CommandInvoker updates, it will go through all the commands and execute them, then move them to the history buffer. The InvertMovement command will just reference the CameraController, and switch the InvertInput bool. It will also store the previous state of that bool to be used for undo/redo (incase some sort of desync happens after some time since initial call, from the game itself doing something).

**Management System**

Observer Pattern. Would observe player shoot input and relay that information to the enemies (same example given in practical). Would have an Observer class that holds all subjects in a private list, a public function NotifyAll which calls the Notify function on each subject, a public AddSubject function which adds a subject to the private list.