Solo Project

Goal
Practice game development by *recreating a part of a classic game*.

Objectives
- Practice good software design
- Focus on test-driven development
- Create a paper prototype

Description
Choose a classic game. Your job in this project is to see how much of it you can create and iterate upon using Unity and C#. Your project does not have to be a copy or a clone of your chosen game, but you should use that game as your starting point. Keep your scope in check, but don't go TOO small… (*If you choose Pong, I expect to see a heavily iterated version of Pong…*)

Plan it in pieces. Make a few, key objects, then add the fundamental mechanics. Then add features piece by piece, testing them as you go. You don't need to have a full, complete game, but you should have an essentially complete prototype/demo. That means a playable game, with some graphics, sound, music, GUI, etc, but perhaps only a single level, or a small number of enemies or collectables. If you are unsure of what you should do, please reach out to me.

Design Tips
1. Remember your software engineering. Keep your objects *loosely coupled* and *abstract anything that can be abstracted*. For instance, if you were making a 1-player Pong with a Pong AI, you could have a Game Object for each of them, and then each of them would have their own script: Player and AIPlayer respectively. However, you may find it easier to have an abstract behavior for the script and control it separately. In other words, you would have a MovableObject script with methods for moving around in both objects, and then Player and AIPlayer controller objects that reference and then call the MovableObject parts of those objects. At first glance, this may seem to be a long version with the same result, but the second design is more forward-thinking. If later you want to add a second player, all you would need is a version of the Player object. You could also easily have multiple types of AIPlayers (for difficulty levels, or types of enemies) or move the object by alternative input devices! (Think cross-platform development).
2. *Test-driven development* says to design tests first, and then fit your code into them. Try to keep your code working, and iterate through it. (The first version of your logic will look terrible! Everyone's does, but the final version should not). In game terms, this means to plan how you want your game to play and

then build around that. See the *MDA* section of the text. Think about what you want your player to experience, then how they will experience it, and then how you will implement it.

3. A paper prototype is a physical, analog prototype of your game. (*The game exists in cyberspace, the paper proto in meatspace*!) For example, if you were going to create a top-down adventure game, you would cut out a square with a picture of your player character on it, cut out a few more pieces for collectables or enemies, draw out a map, and then move them around. This is the game development version of pseudocode. It will help you refine mechanics or experience portions of your game before you sit down to code them. *See chapter 9 for more details.*

Deliverables

1. Your paper prototype won't submit well over Blackboard, so instead, write up a short Word document describing your prototype with plenty of photographs and maybe some diagrams if applicable. *This is due separately and before the rest of the assignment.*
2. Unity project in zip form. (*Assets and ProjectSettings only*). - *for grading*
3. Unity project in .exe or .app form. - *for easy playtesting*
4. Unity project in web form. - *for easy sharing*
5. A short instruction manual in pdf form. It should set up your game (*aesthetics*!), and describe how the game works and how to play (*mechanics*! *dynamics*!!).
6. A project post-mortem. This is a brief document that describes your development process. You should talk about why you chose your specific game, what your plans were as far as changing it, how you approached its development, what worked, what didn't, and any other relevant information regarding the development of the project. *See the References on Blackboard for examples.*