Я смог посмотреть только код, сам проект и его структуру в Unity я к сожалению не видел. Так что мой фидбэк распространяется только на скрипты.

В принципе, всё что я увидел было довольно добротно, видно что вы уже многому научились и движетесь в правильном направлении.
В частности, очень хорошо что для каждой функции вы пишите XML-доки (даже профессионалы с большим опытом это делают не всегда). Я к сожалению не смог прочитать что конкретно вы пишите, но это проблема моего немецкого Макбука, который не смог распознать кодировку кириллицы, а не ваша.
Хорошо, что вы умеете пользоваться Юнити эвентами - это позволяет сильно уменьшить зависимость объектов друг от друга.
У меня не много советов по увиденному коду. Главный: если вы выбрали стиль названий переменных, пожалуйста придерживайтесь его. То есть, если вы решили, что названия сериализированных переменных должны начинаться с маленькой буквы, то они все должны начинаться с маленькой буквы а не с большой или с подчеркивания. Это сильно улучшит читабельность кода.
Постарайтесь свести код в Update к минимуму, особенно не используйте такие интенсивные методы как Vector3.Distance (здесь используется квадратный корень, очень медленная функция, в Update если нет выбора, лучше использовать SqrMagnitude - она делает тоже самое, но без корня, а результат можно сравнивать с квадратом величины - квадрат это всего лишь умножение, одна из самых быстрых операций доступных процессору). Так же в Update не стоит использовать GetComponent<> - это тоже очень медленная функция. Её нужно стараться использовать только один раз, а результат кешировать. Вообще, почти всегда есть возможность избежать использования Update - через эвенты, корутины, DoTween...
Больше я ничего серьёзного не заметил. Поэтому постараюсь дать вам пару советов для дальнейшего развития вашей карьеры.

- Всегда думайте, что ваш код может увидеть кто-то другой. Это может случиться в любой момент - кто-то увидит ваш проект и предложит его купит, или вы устраиваетесь на работу, попросят примеры работ, а на диске только проекты писаные "под себя". Ваш код уже сейчас довольно аккуратен, к этому я бы посоветовал давать функциям и переменным более "описательные" названия. Не ленитесь писать длинные названия - компайлеру это всё равно, но это может помочь вам если вы вернётесь к этому коду через пару месяцев - или кто то другой должен будет продолжать с этим кодом работать (на профессиональном жаргоне это называется "self-documenting code" - многие программисты считают, что поэтому писать XML доки не имеет смысла - но это полемика..)
- Если вы не планируете работать только в русскоязычных фирмах, постарайтесь подтянуть английский. Это вам откроет доступ к 99,99% информации в интернете и значительно повысит шансы трудоустройства. В частности я бы посоветовал даже комментарии в частных проектах писать на английском - опять же, если этот код получит кто-то другой, кто возможно не говорит по русски. Мне приходилось работать с кодом комментированным на польском, японском и турецком - было сложно:) Не стесняйтесь уровня своего английского - единственный работающий метод выучить язык - это начать им пользоваться. Всё остальное быстро придёт с практикой. Если вы интроверт (как я) много

читайте и слушайте, потом пытайтесь разговаривать с людьми. Если вы экстраверт - начинайте говорить сразу:)

- В этом простом проекте я не ожидал какой-то особой архитектуры или паттернов. Поэтому я вам просто подскажу какие методы сейчас популярны:
    - Command Pattern
    - Factory Pattern
    - Flyweight Pattern
    - State Machine (важно)
    - Strategy Pattern
    - Dependency Injection (мнения об этом паттерне неоднозначны, но многие проекты которые его используют)
    - Global Messaging System (мой личный фаворит - позволяет объектам быть не связанными с друг другом, что решает проблему "спагетти" кода и повышает скалируемость системы)

К Singleton отношение очень не однозначно. Считается, что он поддерживает "ленивый" стиль программирования и тот самый "спагетти" код. Он в принципе используется, но в очень особых случаях, и его почти всегда можно избежать.

Вообще, вопрос о паттернах почти религиозный - часто программисты считают тот паттерн "единственно правильным", который они хорошо знают и им комфортно с ним работать. Я считаю, что правильных нет, есть подходящие проекту. В личных проектах - вам решать какая архитектура вам удобна. Если вы работаете в команде - тогда нужно конечно подстраиваться под стиль других. Или если у старшего программиста есть любимый паттерн - тут уж не поспоришь. Но и вы когда-нибуть будите старшим программистом.

В конце я хочу сказать, что я вижу у вас много энтузиазма, прилежания и "голода" на знания и навыки, поэтому я считаю что у вас есть большой потенциал. Я надеюсь вы многого добьётесь как программист игр.