**По текстовому заданию:**

* Хорошие примерчики, вопросов нет) Разве что для ворот я бы машину состояний не городил)

**По заданию с NPC:**

* Немножко странно в плане того, что задание было реализовать NPC, а не игрока, но ладно
* Совершенно ненужное поле StateShow и StateName (засоряет код)
* Также очень неаккуратно по код стайлу в плане отступов, пустых строк и тп – код выглядит довольно неряшливо
* Также название BasePlyaerClass – не несет в себе какого – то смысла (по сути тут может быт все, что угодно)
* В целом принцип работы машины состояний не такой как описывалось в задании, но пойдет для отработки – поработал с инпутом и в целом переходы между состояниями тоже сделаны, так что окей
* Вообще, стоило попробовать реализовать некоторые попытки отсчета времени или сменяемость цели, чтобы, возможно встретить пару косяков с дубляжом кода, которые встречаются у всех почти.
* В качестве доп примеров можешь посмотреть тоже хорошие реализации этого задания для NPC:
  + <https://gitlab.com/boykopb/patterns_course/-/tree/main/patterns_course/Assets/Week_2/HW_2/Scripts>
  + И один из участников интересно придумал сделать это задание, реализовав некоторую машину состояний с памятью <https://github.com/kostegs/Lesson2_State_Task2/tree/main/Assets/Scripts>

**По заданию с добавлением стейтов бега:**

* Вообще сделано неплохо, но тут такая же ошибка, как у большинства (поэтому это задание я и даю😊). У тебя есть несколько состояний бега, но почему – то связанные с этим делом зажатия кнопок и тп обрабатываются в Grounded стейте (в целом решение нормальное, но это пока у тебя мало состояний наземных и логика не такая сложная)
* Я бы выделил бы стейт MoveState и от него бы реализовал уже три стейта бега, т.е. базовый бы как раз контролировал все, что связано с бегом, а так щас получается, что у тебя это делает GroundedState, что не совсем логично. Т.е. тут хотелось бы еще больше отработать иерархическую модель и выделять логически стейты, ведь если ты добавишь базовый MoveState, то тебе даже трогать Grounded стейт не придется, ты просто добавляешь поверх новую логику – в этом и прелесть машины состояний
* В качестве пример реализации, которая мне нравится могу посоветовать вот эту (тут реализация с зажатием для бега + правильно разбиты состояния) - <https://github.com/PetkaKahin/Lesson2/tree/main/Assets/CharacterExample/Scripts/Character/StateMachine/States/Grounded>
* В остальном все хорошо.С инпутом тоже можно по разному реализацию делать, в целом у тебя неплохой вариант. В примере выше, немножко по-другому реализовано, тоже можешь посмотреть для кругозора)

**По медатору:**

* Очень плохо проверять по ИМЕНИ ОБЪЕКТА ЧТО-ТО. Это крайне нандежно, можно по компоненту, в крайнем случае по слою или тегу, но это тоже плохо. Кто-то поменяет название и вес полетит + не вынесены сами названия в какие-то константы
* Также очень неаккуратно по код стайлу
* В медиаторе тоже косяк – я несколько раз проговаривал, что в нем недолжно быть НИКАКОЙ ЛОГИКИ ИГРОВОЙ. Тут же идет проверка хп, что является частью игровой логики. Лучше сделать событие смерти, как вариант. Медатор должен только связывать, но не заменять сущности собой частично – иначе мы жестко привязываемся к этому медиатору, а хотим наоборот отказаться от него
* Также я бы для игрока медиатор напрямую, возможно не передавал (плюс у тебя медиатор все равно на события подписывается игрока, так что какой смысл в этом)
* И еще момент с UIClass – название ни о чем не говорит, но, в целом, сам класс для примера устроен нормально.
* RSTPressed – тоже название такое себе – лучше полностью называть)
* В остальном все неплохо, в целом медиатор связывает это дело все, но вот момент с логикой в нем - опасный