**Базовые синтаксические правила.**

При написании кода придерживаемся базовых синтаксических правил MSDS.

1. Наименования классов с большой буквы каждое слово, слитно - SomeClass
2. Наименование структур с большой буквы каждое слово, слитно – SomeStruct
3. Наименование публичных переменных с большой буквы каждое слово, слитно – MyParam
4. Наименование приватных переменных начинаются с прочерка - \_myParam
5. Намименование локальных переменных метода с маленькой буквы без прочерка – myParam

При написании кода и наименовании переменных MonoBechavior придерживаемся нижеуказанного стиля:

1. Наименование ScriptbleObject класов перед названием используем приставку Scr\_ - Scr\_WeaponInfo
2. Наименование UI элементов всегда начинается с первой буквы их класса:
   1. **private** Button b\_**e**xitGame;
   2. private Image i\_photo;
   3. ptivate TMP\_Text t\_errorText;
   4. ИСКЛЮЧЕНИЕ InputField if\_name;
   5. **public** Button b\_**E**xitGame;
3. Наименование переменных Coroutine всегда должно заканчиваться словом Coroutine – myCoroutine
4. Наименование интерфейсов всегда начинается с заглавной буквы I – Iwindows

**Правила написания кода и его структуры**

1. Код должен быть понятным и легко читаться другими участниками процесса разработки.
2. Придерживаемся системы блоков, то есть когда пишите код, старайтесь писать его так, что бы его можно было вырвать из проекта и закинуть в другой проект, при этом он будет работать без изменений.
3. Помните, что **вы пишите код не для себя, а для команды**, и чаще всего **с классами в UnityEditor будете работать не вы, а левелдизайнеры** и геймдизайнеры, а также другие специалисты. Пишите код таким образом, чтобы другие участники команды, не связанные с программированием, могли легко разобраться что делает ваш класс и как с ним работать в едиторе. + с написанным вами классом должно быть в принципе удобно работать в Editor-е юнити.
   1. Старайтесь избегать вывода в общий доступ переменных, которые не нужны левелдизайнерам.
   2. Старайтесь использовать подписи для переменных [Header(“Кнопка выхода из игры”)]
   3. Старайтесь делить блоки переменных линиями или переходами используя все тот же Header или другие системы удобства вывода в едитор.
   4. По возможности старайтесь уменьшать количество необходимых связей, которые нужно будет устанавливать в Editore, делая это сразу в коде, если это возможно. (Пример: кнопку можно подписать на действие в коде, тем самым делая код более контролируемым и более гибким)
   5. Если точно знаете, что на объекте должно быть твердое тело, используйте RequestType для автосоздания данного компонента на объект. НО ОЧЕНЬ ВНИМАТЕЛЬНО!!!
   6. По возможности, если написанный класс будет использоваться при создании уровня, пишите к нему инструкцию, в идеале со скринами сцены, что бы человек мог сверяться с ней и тратил меньше времени на изучение ваших систем.
   7. Как пример, смотрите как пишут классы, которые вы часто используете разработчики самого движка Unity3d. Например RigidBody
4. НЕ ПОВТОРЯЙТЕСЬ! Если какой-то метод, который вы используете, часто дублируется в разных классах, попробуйте вынести его в статический класс SupportClasses и обращаться через него. Не пишите один и тот же код по многу раз. Так же вы можете использовать интерфейсы или перегрузки абстрактных и обычных классов. Пользуйтесь всеми возможностями современного программирования.
5. Старайтесь не писать в классе лишнего! Это вам не нужно! Придерживайтесь правила, пишем то, что нужно сейчас и думаем о будущем. Возможно к вашему классу в будущем придется дописать дополнение, но сейчас это точно не известно. Соответственно не нужно писать доп. Методы, позволяющие делать какие-то не нужные сейчас действия, НО старайтесь писать общую структуру нынешнего кода так, чтобы в дальнейшем с ней было легко работать и использовать ее при необходимости создания доп. Функционала в вашем классе и ли с его использованием.
6. Придерживаемся принципа инкапсуляции. Не нужно плодить публичные переменные. Все публичные переменные, если это возможно должны быть инкапсулированы, то есть стать приватными с доступом к ним только через метод или другую публичную переменную. В идеале, переменная может читаться, но не должна изменяться извне.
   1. Пример (

private TMP\_Text t\_text;

public string GetTextValue()

{ return t\_text.text; }

public void SetTextValue(string newText)

{ t\_text.text = newText; }

)

1. Одно из базовых правил архитектуры: Есть менеджер, есть контроллер. Контроллер отправляет запрос на действие менеджеру. Менеджер принимает решение, делать или нет, после чего менеджер отправляет команду контроллеру (тому-же самому или другому на произведение действия)