Описание ключевых классов и их взаимодействия

1. SceneContex - служит для последовательной инициализации классов - Monobehaviour на сцене. А также для инъекции ConfigProvider в эти классы. Такие решения были приняты в ходе имплементации, так как инициализация из одной точки входа упрощает отладку и дает больше контроля там порядком выполнения методов на старте игры. Для инъекции ConfigProvider можно было выбрать Zenject, но для тестового проекта я принял решения использовть инъекцию через метод инициализации.
2. SceneObject - базовый класс для используемых Monobehavior который содержит ссылку на ConfigProvider и виртуальный метод инициализации, который нам позволяет добавлять дополнительные действия в ходе инициализации в дочерних классах.
3. ConfigProvider - ScriptableObject который содержит базовые настройки, которые могут меняться в рантайме. Например очень удобно было реализовать изменение скорости. Так как все элементы (монетки и платформы) имеют инжектованый доступ к этому изменяемому значению. Главное не забывать, что скриптабл обжекты сохраняют измененные значения после запуска, для этого был добавлен метод резета скорости. Но в более глобальном проекте, я бы создавал инстансы в рантайме для таких объектов (есть опыт построения архитектуры на скриптабл обджектах). Также сюда я добавил события со стартом и концом действия эффектов. Возможно их стоило выделить в другой класс, но так как доступ к этому объекту был инжектирован, то на эти события можно было легко подписать прямо во время инициализации.
4. Effect - базовый абстрактный класс для всех эффектов. Выбрал абстрактный класс, а не интерфейс, так как заметил, что некоторые методы (Apply, и FinishAction) одинаковы для любых эффектов, но оставил возможность их переопределения.
5. SpeedUpEffect, SpeedDownEffect и FlyEffect - наследуемые классы от Effect. Тут можно отметить, что новые эффекты будет достаточно легко добавить, достаточно создать новый класс унаследованный от Effect и часть функционала уже готова.
6. EffectPerformer - реализации управления временем для всех эффектов. Сделано через Monobehaviour так как в таком случае можно воспользоваться корутинами и методом Update. Тут логика была сделана так, что если какой-то эффект уже действует, и мы уже вызываем новый, то текущий досрочно завершается. Однако можно переписать метод PerformEffect таким образом, чтобы например эффекты накладывались.
7. Coin - логическая составляющая часть монетки, которая содержит свой эффект (принимает его через конструктор), и метод подбора монетки, который соответственно и активирует соответствующий эффект.
8. CoinsFabric - фабрика разных типов монет с соответствующими эффектами. При добавлении нового типа монетки/подбираемого объекта, тут нужно будет дописать метод по его созданию.
9. CoinView - представление логической монетки уже непосредственно на игровой сцене. Добавил сюда перечисление из типов эффектов, так как подумал, что дизайнерам было бы удобно выбирать тип эффекта/монетки прямо в префабе. И дальше при инициализации через фабрику создается соответствующая логическая часть. Могу отметить, что тут можно было реализовать наоборот от логики (например передавать логическую монетку в инициализацию при создании визуальной части). Но тут зависит как будет удобнее на проекте.Также здесь происходит ее движение и обработка столкновений: с игроком - через логическую часть вызываем соответствующий эффект. И в любом случае отключаем объект (для дальнейшей реализации пула объектов.
10. CoinsSpawner - этот класс и отвечает за спавн этих CoinsView. В каких местах, и как часто они будут появлятся. Также здесь реализован вышеупомянутый пул объектов для оптимизации. Мы не создаем объекты бесконечно, а также не удаляем их, а переиспользуем.
11. PlatformsGear - класс, отвечающий за бесконечное перемещение платформ. В проекте я оставил всего 2 платформы, но при надобности туда можно добавлять и больше префабов-платформ, главное, чтобы они были одинаковой длины (в данной реализации).
12. PlayerMovement - тут реализованы перемещение игрока между дорожками свайпом, смена анимации и полет игрока.

И прикрепляю упрощенную сцену классов:

