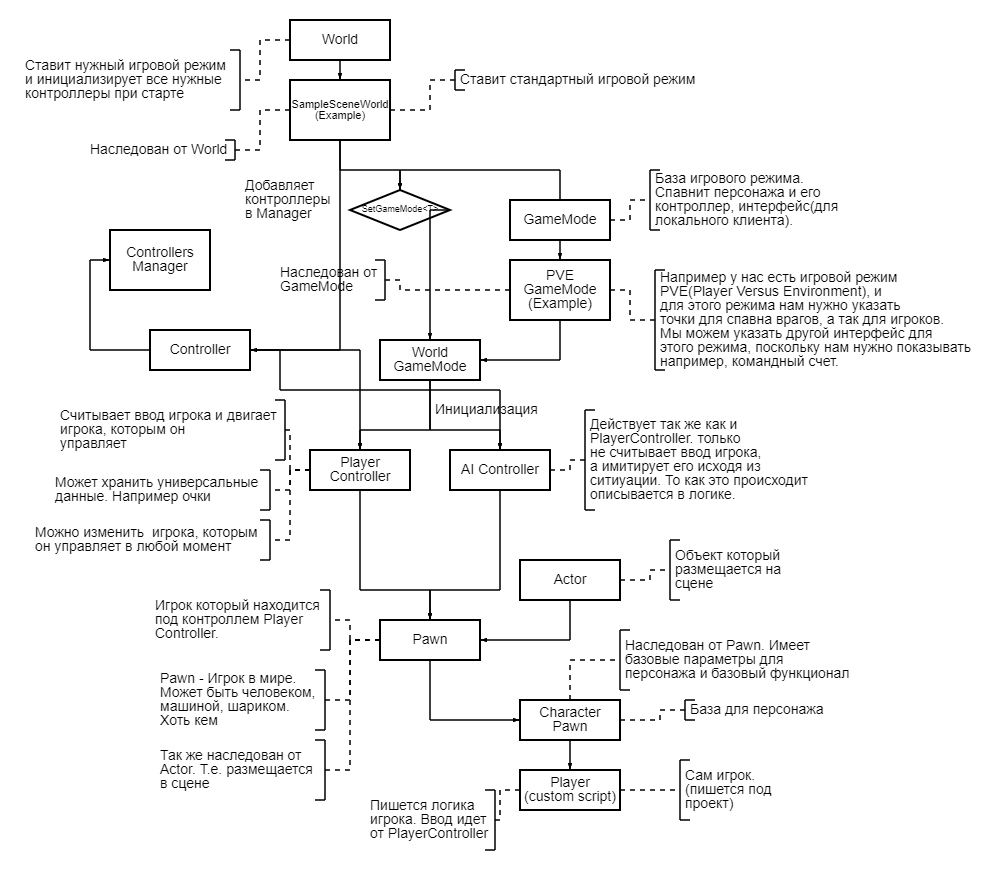
**Вступление**

Зравствуй. Я не мастер писать какие-либо технические документации, но думаю ты сможешь понять основу работы этой архитектуры. Тут не будет слишком много подробностей о там как она работает изнутри. Однако вкратце нужно будет об этом поведать.



Примерно так она выглядит. Страшно, не правда ли? Она становится довольно понятной если разбирать её по немного и приводить примеры связей между блоками.

Компонент World

Пара слов о компоненте

Пожалуй сперва ознакомимся с ним. Поскольку он по своей сути является инициализатором всего. Он помещается на пустышку(пустой GameObject) в сцене и в настройках компонента ставится GameModeData(поговорим позже).

В его обязанности входит:

1. Поставить игровой режим для мира.
2. Создать все нужные контроллеры для него.

Сам класс является абстрактным и не может на прямую поместиться в сцену. В идеале для каждой сцены создается новый мир. Он может наследоваться от основного класса мира, или же от вашего(никто не запрещает).

Как использовать?

| public class SampleSceneWorld : World *//Наш мир наследован от World.* {  *//Пример логики для мира*  *//SerializeField нужен для того, чтобы привтные были видны в unity*  [SerializeField] private Trigger \_doorTrigger; *//Триггер с событием.*  [SerializeField] private GameObject \_someObject; *//Какой то объект.*   *//В каждом классе мира нам нужно переопределить этот метод.*  *//В нем мы как раз добавляем все контроллеры и ставим режим.*  protected override void Initialize()  {  AddController(new BallPlayerController());   SetGameMode<StandardGameMode>(); *//Поставим стандартный игровой режим.*  }    private void Start() => \_doorTrigger += OnDoorTriggered; *//При старте*  *//подписываемся на событие.*   private void OnDoorTriggered()  {  \_someObject.SetActive(false);  } } |
| --- |
|  |

Все методы

| Initialize | protected abstract | В нем инициализируются все контроллеры.  Вызывается из базового класса в Awake(). | На вход нету аргументов | void |
| --- | --- | --- | --- | --- |
| SetGameMode | protected | Ставит игровой режим с помощью ControllersManager.SetGameMode<T>(GameModeData) используя переменную с информацией о режиме в базовом классе. | На вход T унаследованное от GameModeBase | void |
| Reload | public static | Чистит все существующие на данный момент контроллеры в ControllersManager и перезапускает сцену | На вход нету аргументов | void |

Класс Pawn

О классе

Pawn - любое представление игрока в сцене. Pawn - абстрактный класс, от него можно только наследоваться. Например можем создать класс Player, который будет наследовать класс Pawn. Этот игрок благодаря Pawn будет игровым и сможет управлятся.

Класс PlayerController

О классе

Является таковым “мостиком” между реальным игроком(человеком(вашим мозгом(результатам работы нейронов))) и игровым персонажем, контролируемого этим контроллером. Может менять игрока во время игры или хранить какую то универсальную информацию. Например очки. В вашей игре может быть игровой режим бой насмерть где игрок будет постоянно умирать и возрождаться. И чтобы не терять его очки, их можно хранить в контроллере.