HOUR 2, LECTURE 3

Understanding the Gameplay Framework

INTRODUCTION

This lecture introduces you to the Gameplay Framework, which is a collection of C++ or Blueprint classes that manages the rules of your game, handles player input, and defines the avatars, cameras, and player HUDs in UE4 projects.

Don’t let the terminology fool you. While UE4 has been developed to make games and much of the terminology is game related, UE4 can be used to make many different applications.

LECTURE GOALS AND OUTCOMES

Goals

The goals of this lecture are to

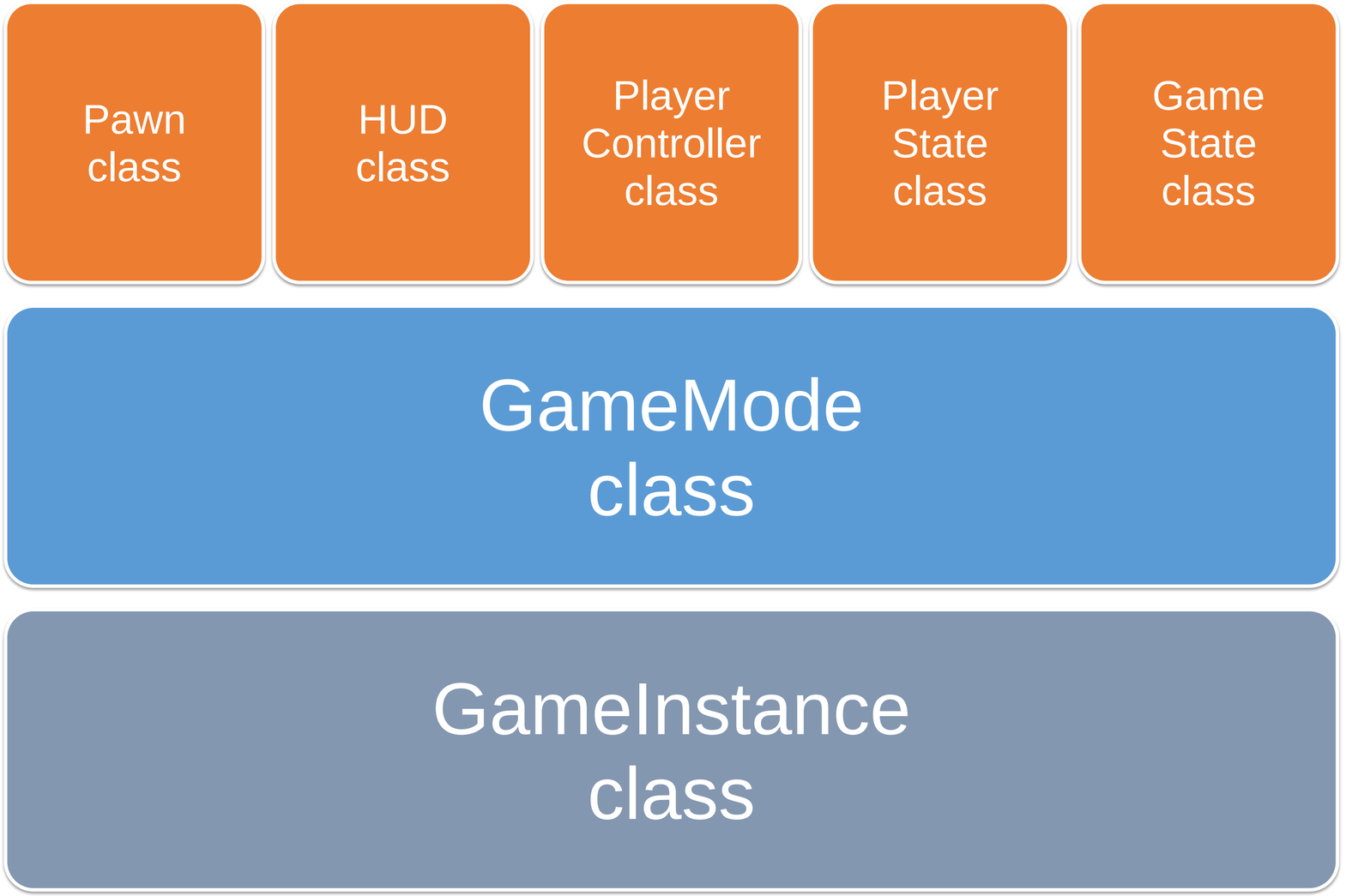
- Become familiar with the Gameplay Framework
- Learn about the relationship between the Game Mode and a Game Instance
- Learn about the Pawn, HUD, Player Controller, Player State, and Game State classes

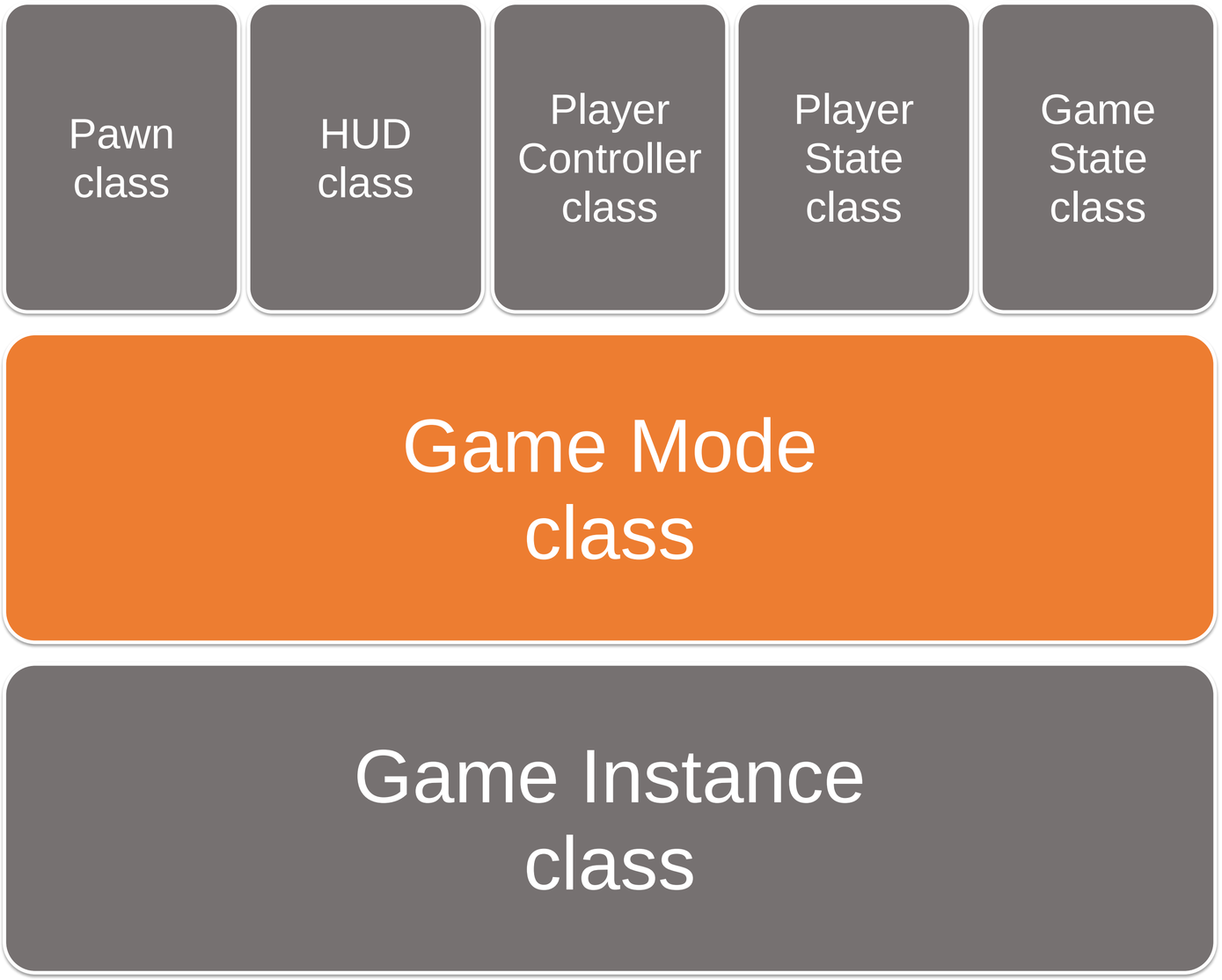
Outcomes

By the end of this lecture you will be able to

- Discuss the Gameplay Framework
- Understand the relationship between the Game Mode and a Game Instance
- Understand the relationship between the Pawn, HUD, Player Controller, Player State, and Game State classes

Gameplay Framework

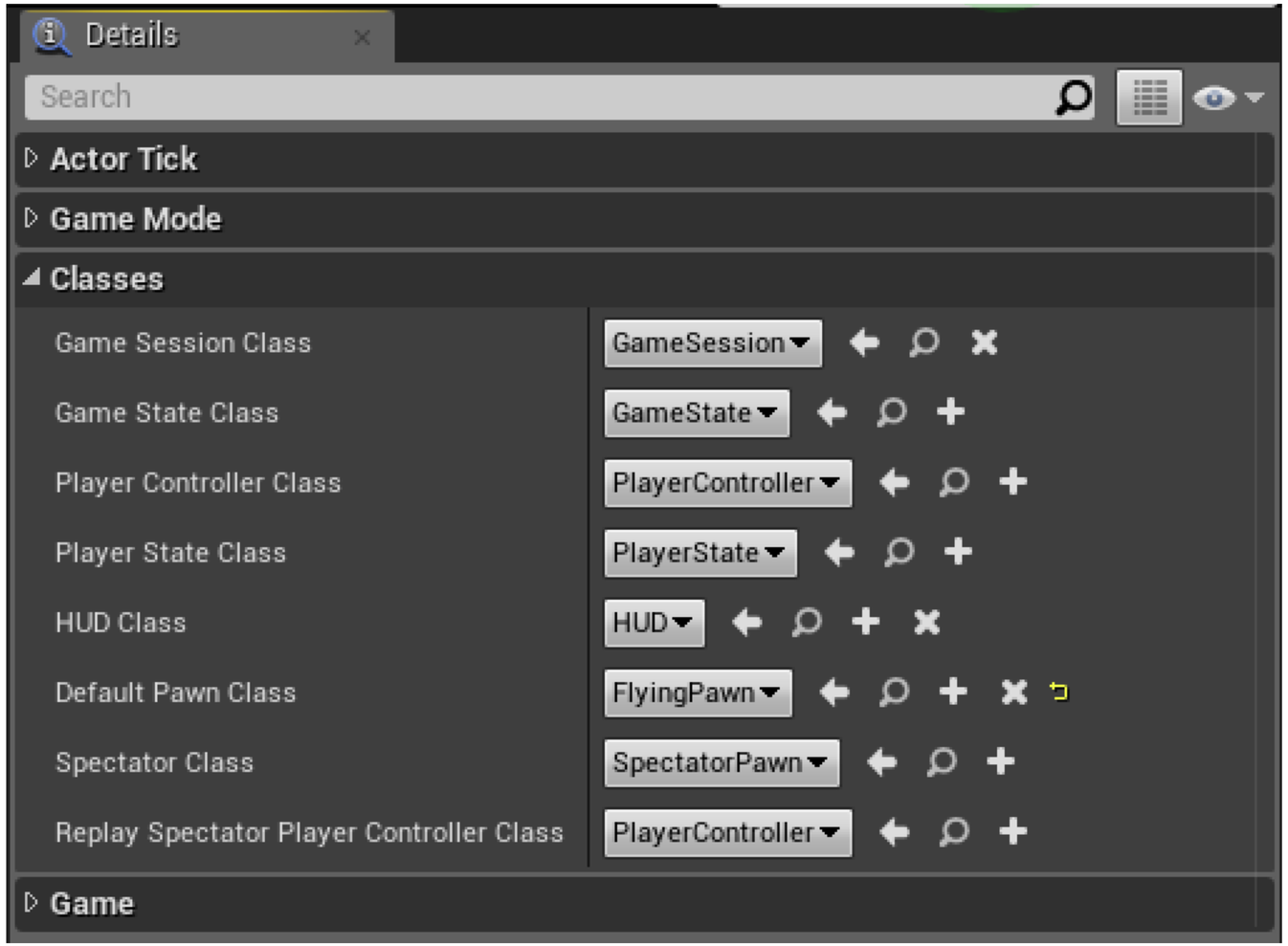




GAME MODE CLASS

The Game Mode class is used to set the rules of the game and store all the other classes that are needed to define the game's core functionality.

***Note**: The Game Mode class is a good place to script respawn systems for a first-person shooter or a timer for a race game.*

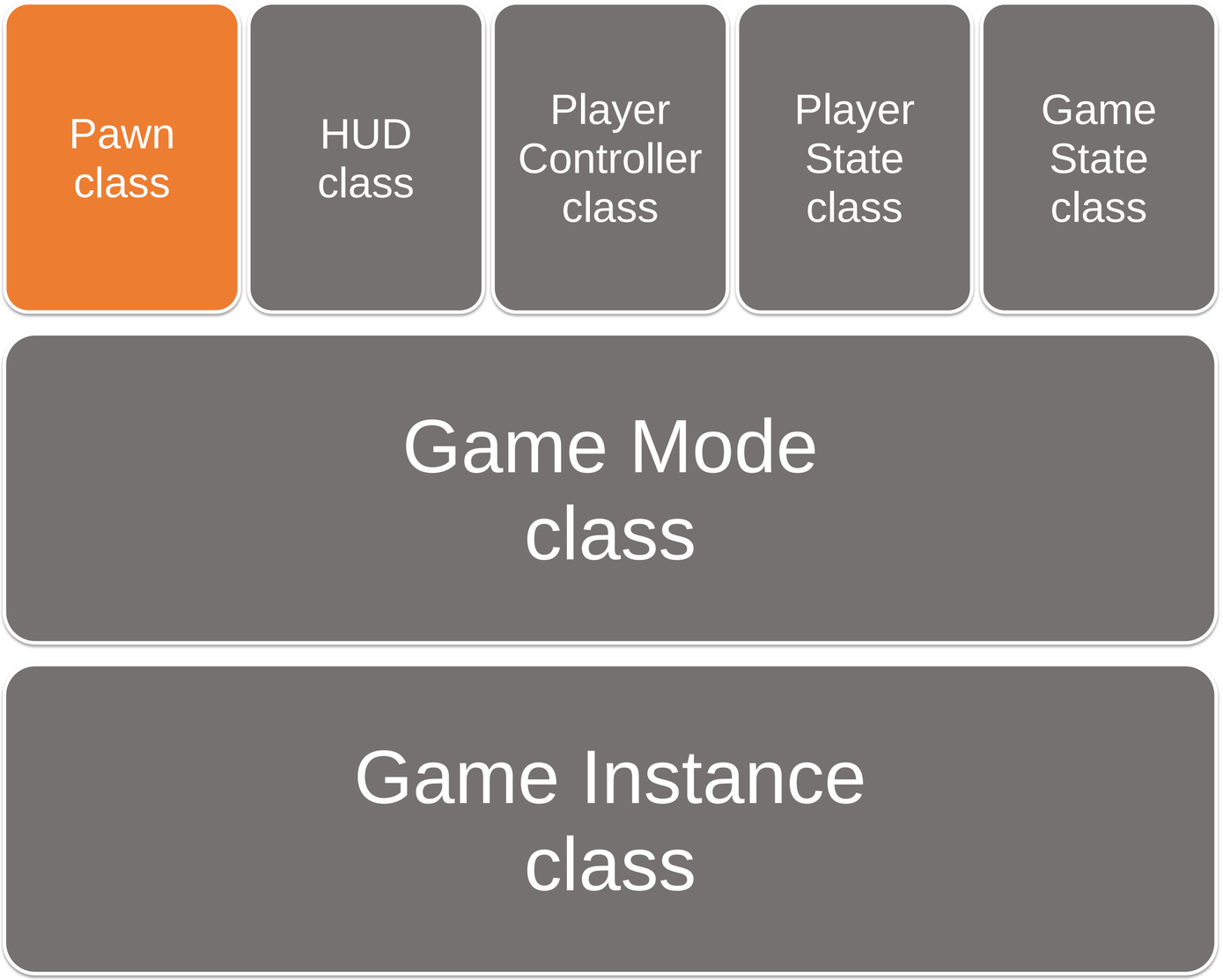


GAME MODE CLASS

You can assign the Game Mode to a project or to each individual Level in a project.

Often, projects have two or three Game Modes, but of course only one can be assigned as the default mode.

Game Modes cannot be changed once a Level is loaded.

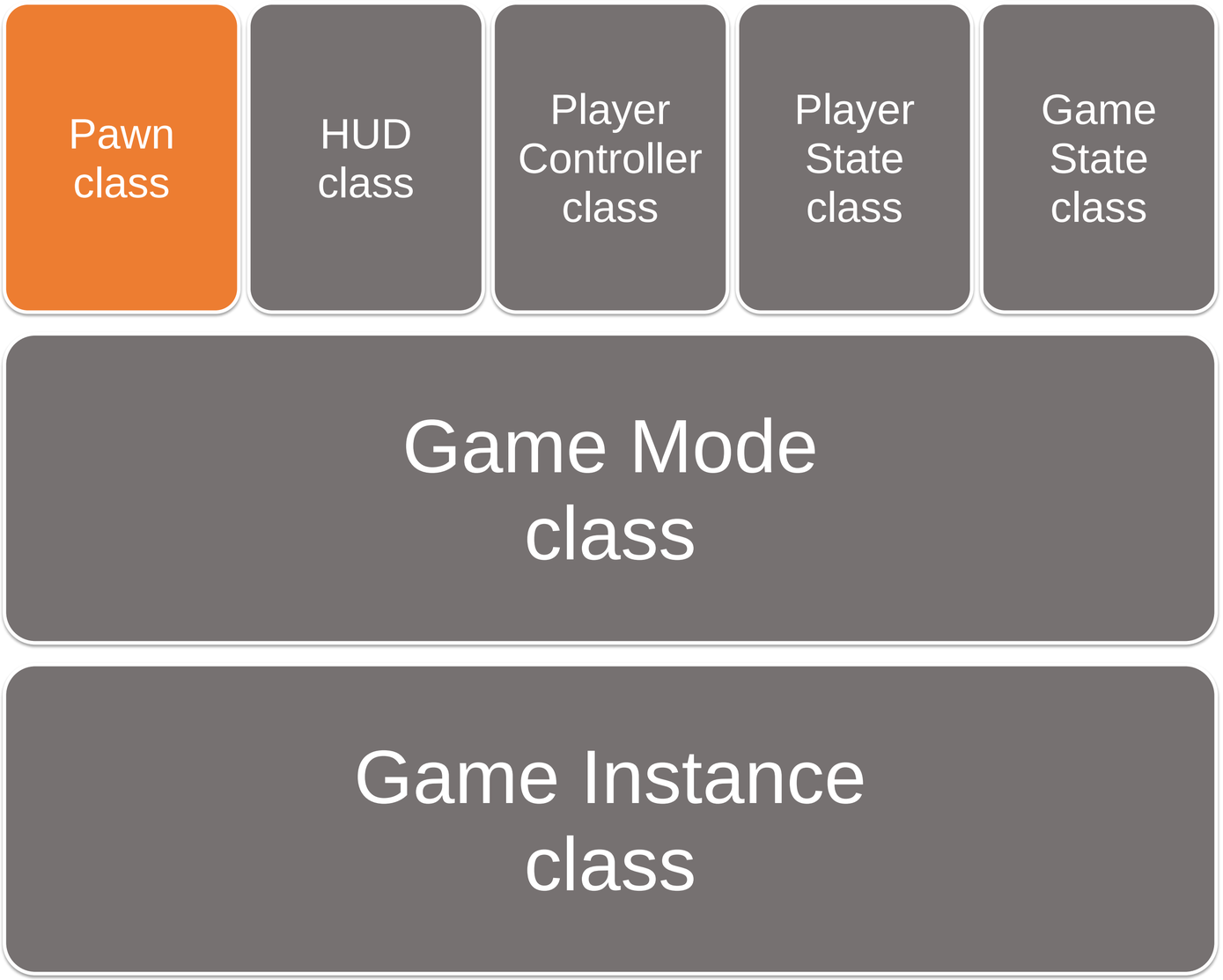


PAWN CLASS

The Pawn **c**lass is the base class of all Actors that can be controlled by players or AI.

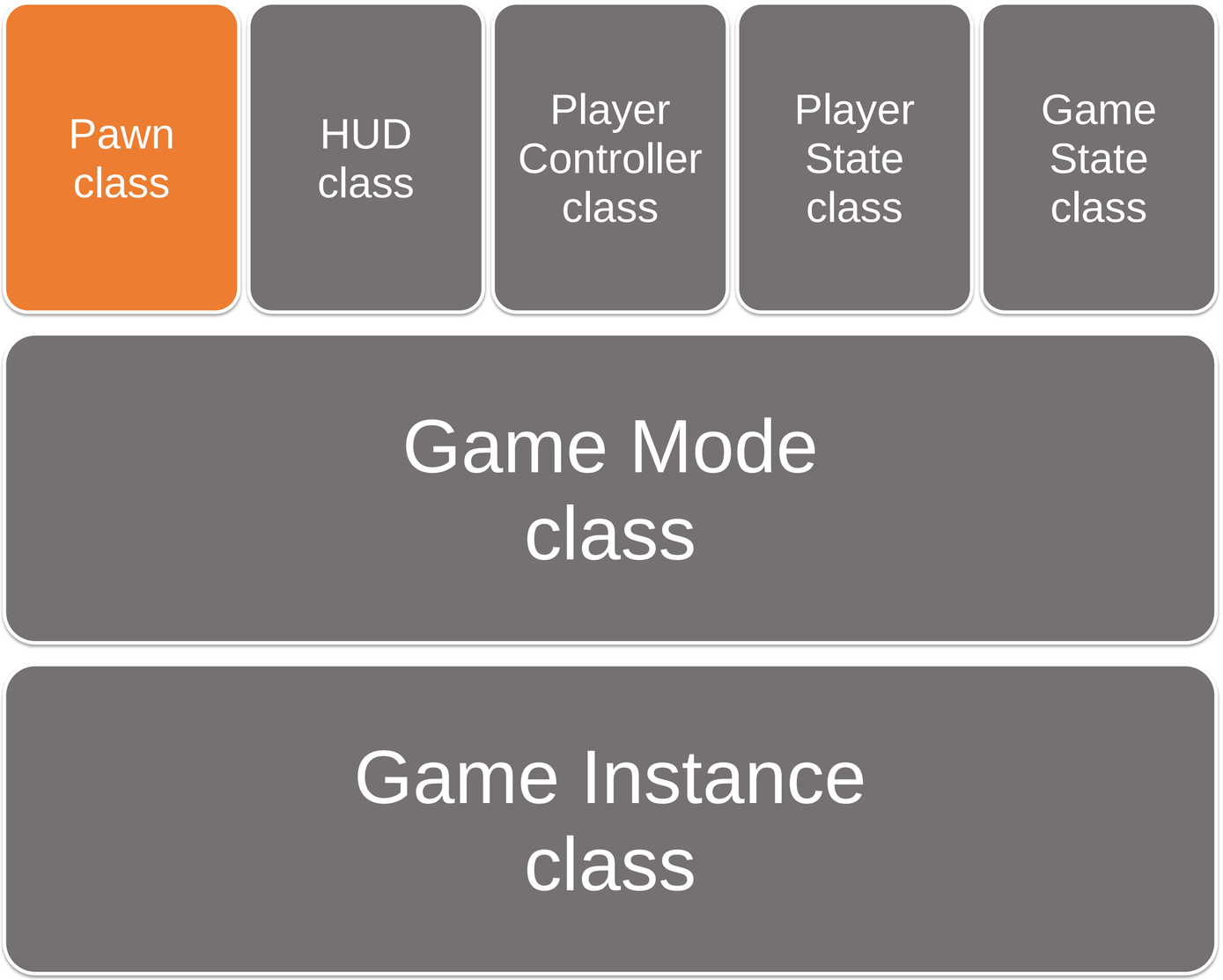
A Pawn is the physical representation of a player or AI entity within the world.

The Pawn determines not only what the player or AI entity looks like visually, but also how it interacts with the world in terms of collisions and other physical interactions.



PAWN CLASS

The Pawn represents the physical location, rotation, and so on of a player or entity within the game. A Character is a special type of Pawn that has prebuilt movement and input systems.

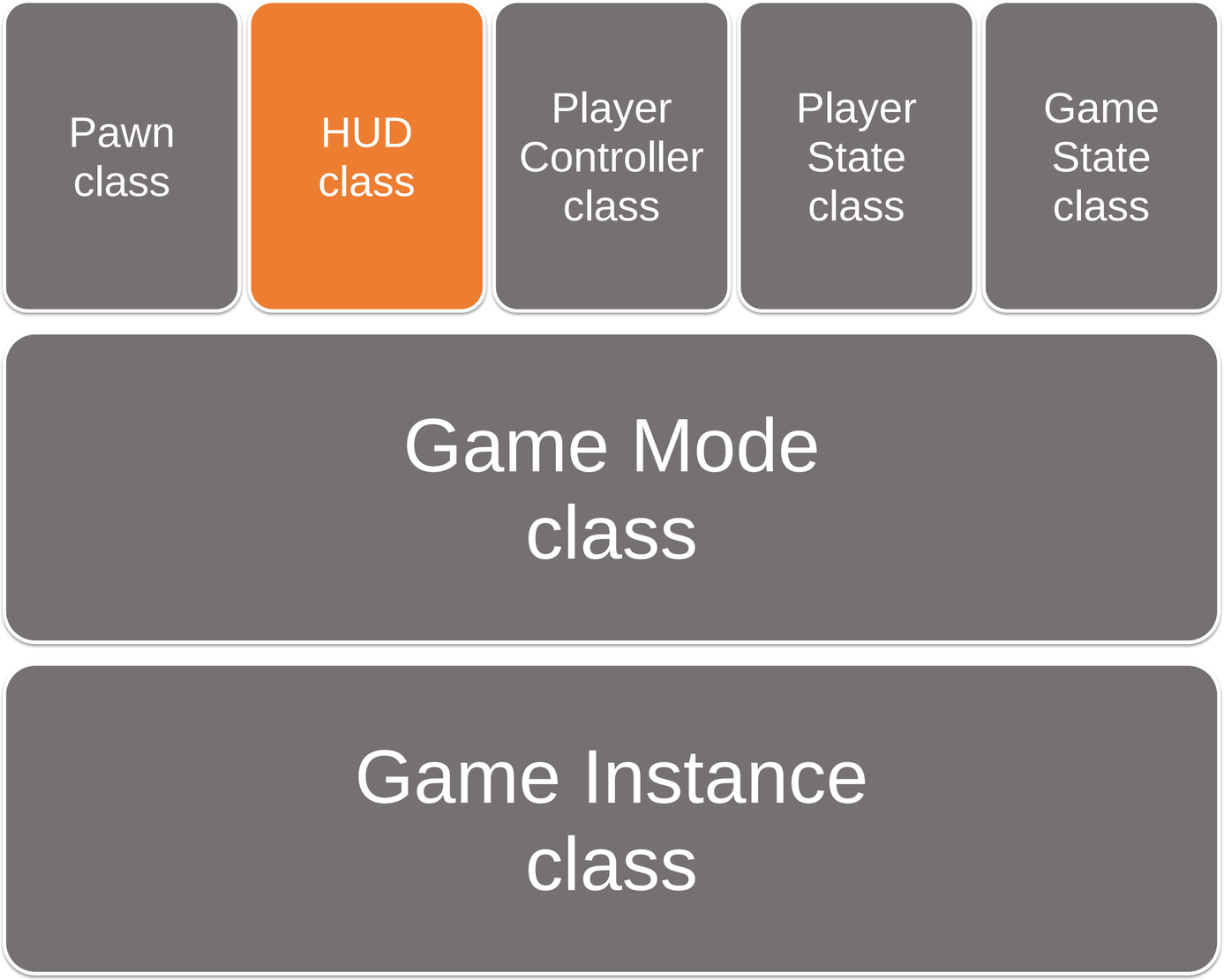


PAWN CLASS

There are several classes that can be assigned to the DefaultPawn class property in a Game Mode:

- Pawn class
- Character class
- Vehicle class

A Pawn class is a generic class for creating a variety of Pawn types, while Character and Vehicle classes are set up for dealing with specific but common Pawns found in most games.



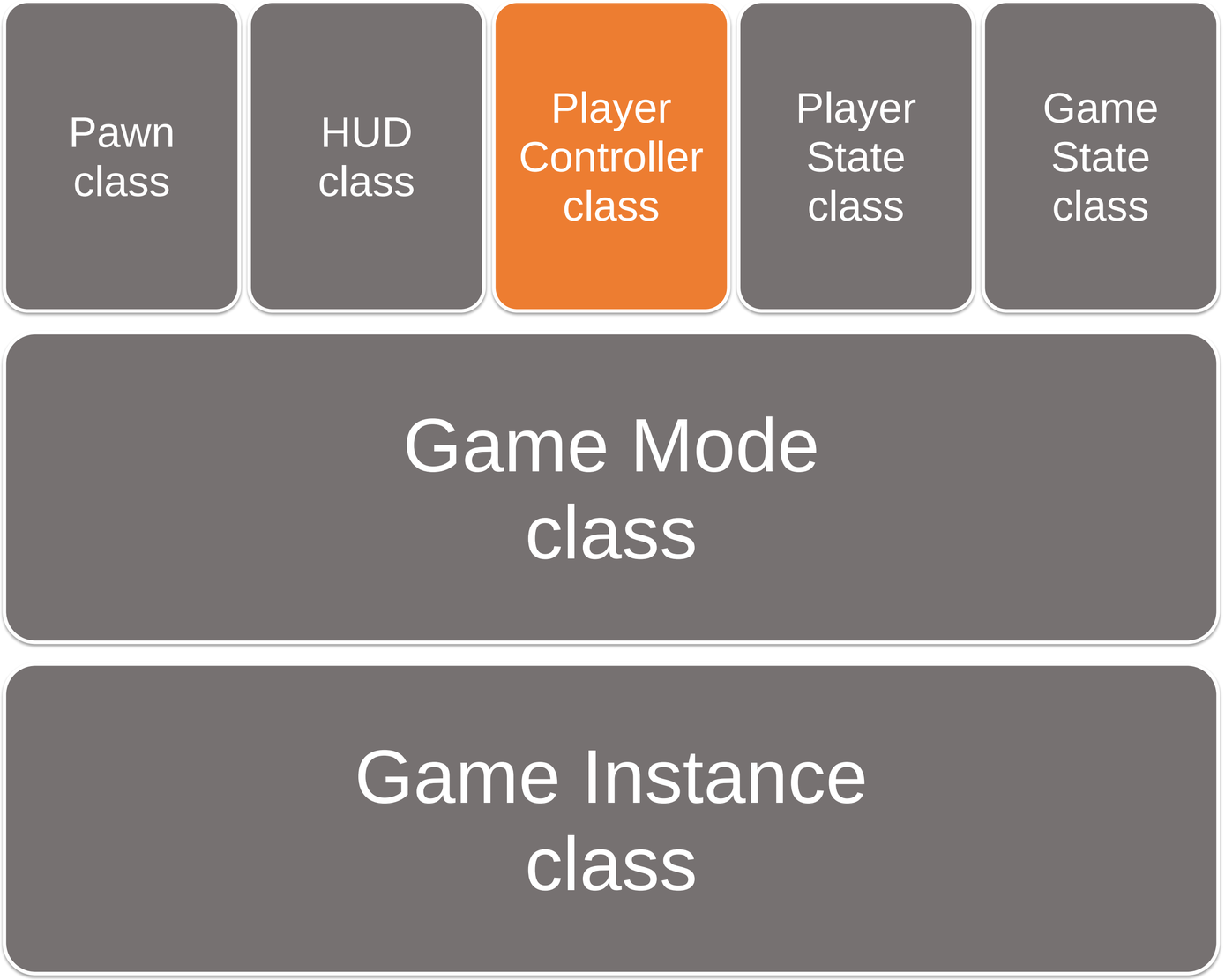
HUD CLASS

The HUD (heads-up display) is the base object for displaying 2D elements overlaid on the screen.

Every human-controlled player in the game has their own instance of the HUD class that draws to their individual Viewport.

In the case of split-screen multiplayer games, multiple Viewports share the same screen but each HUD still draws to its own Viewport.

The type, or class, of HUD to use is specified by the Game Mode being used.

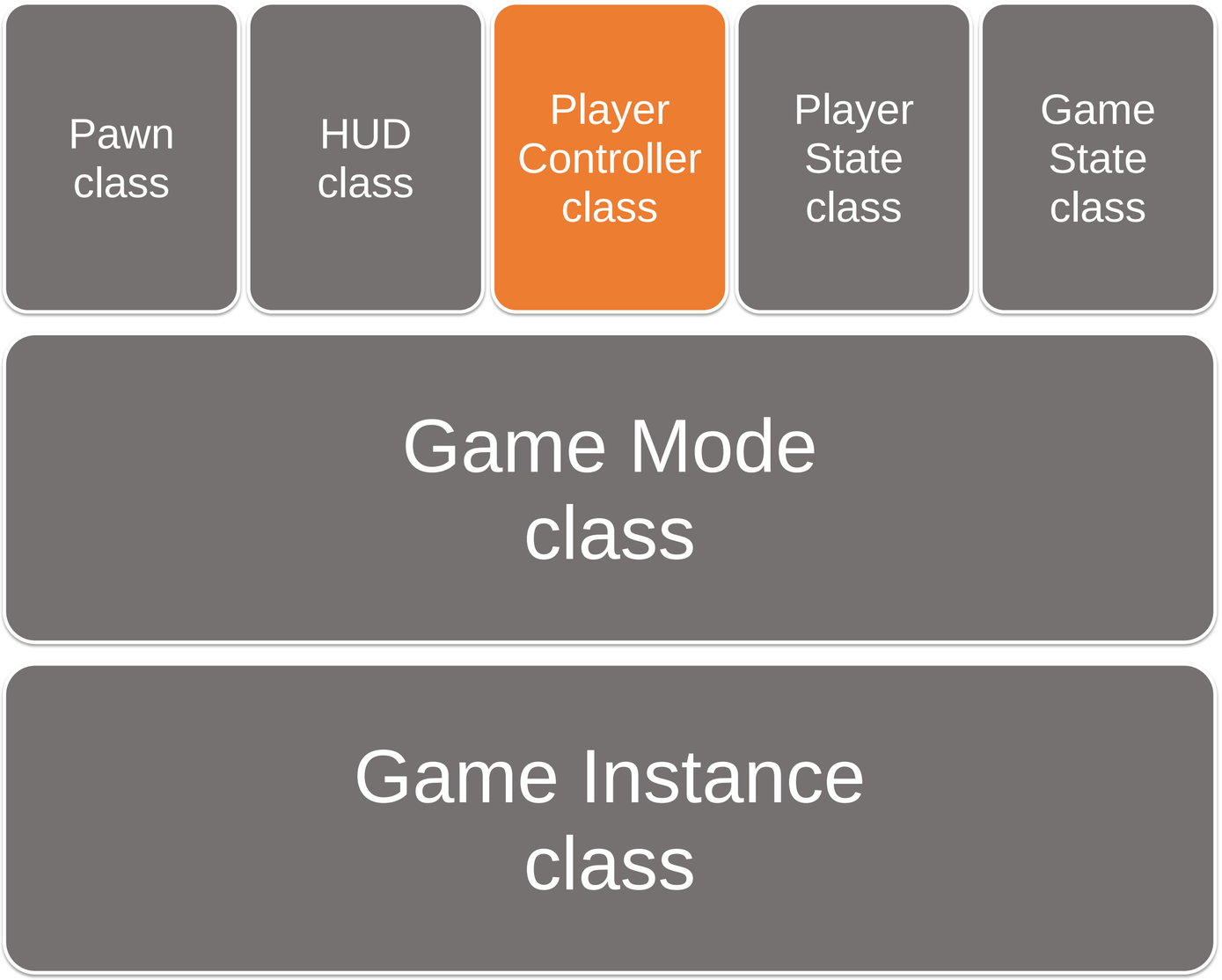


PLAYER CONTROLLER CLASS

A Player Controller is the interface between the Pawn and the human player controlling it.

The Player Controller decides what to do and then issues commands to the Pawn (for example, “start crouching,” “jump”).

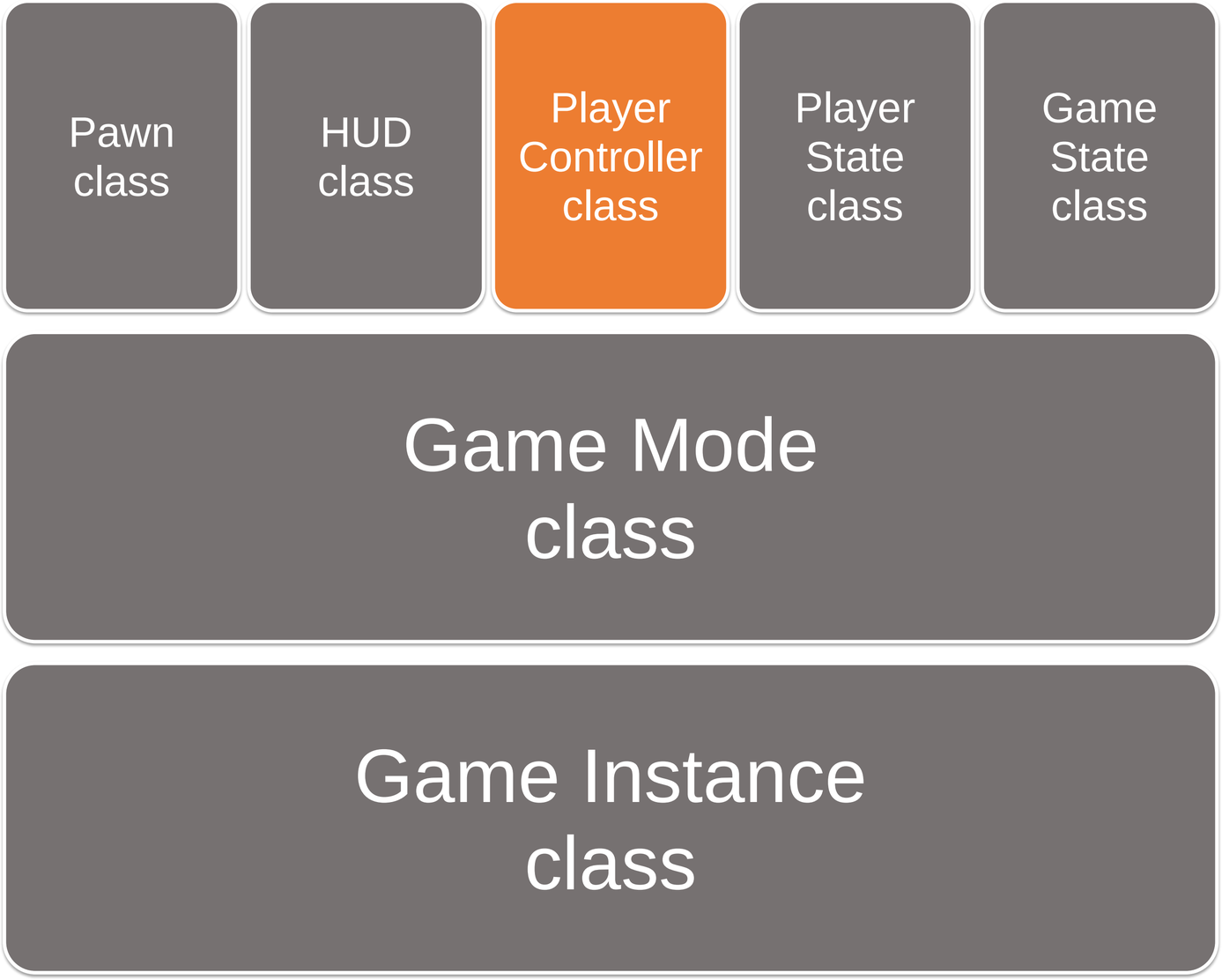
The Player Controller persists throughout the game, while the Pawn can be transient.



PLAYER CONTROLLER CLASS

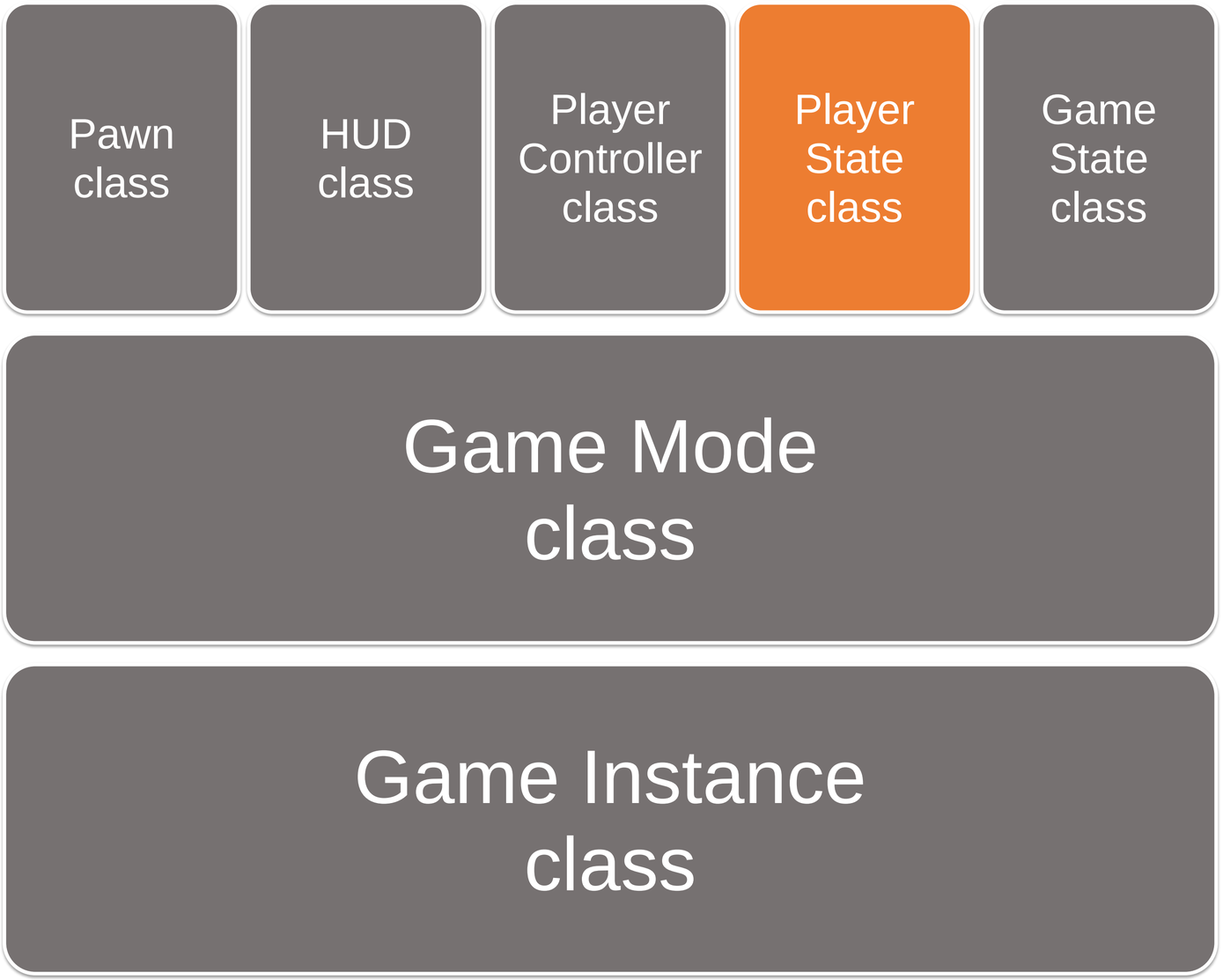
There are two basic types of controller classes: **Player Controller** and **AI Controller**.

Every human player in a game has an instance of the Player Controller class assigned to them.



PLAYER CONTROLLER CLASS

Player inputs can be anything from mice and keyboards or gamepads to touch screens or an Xbox Kinect.



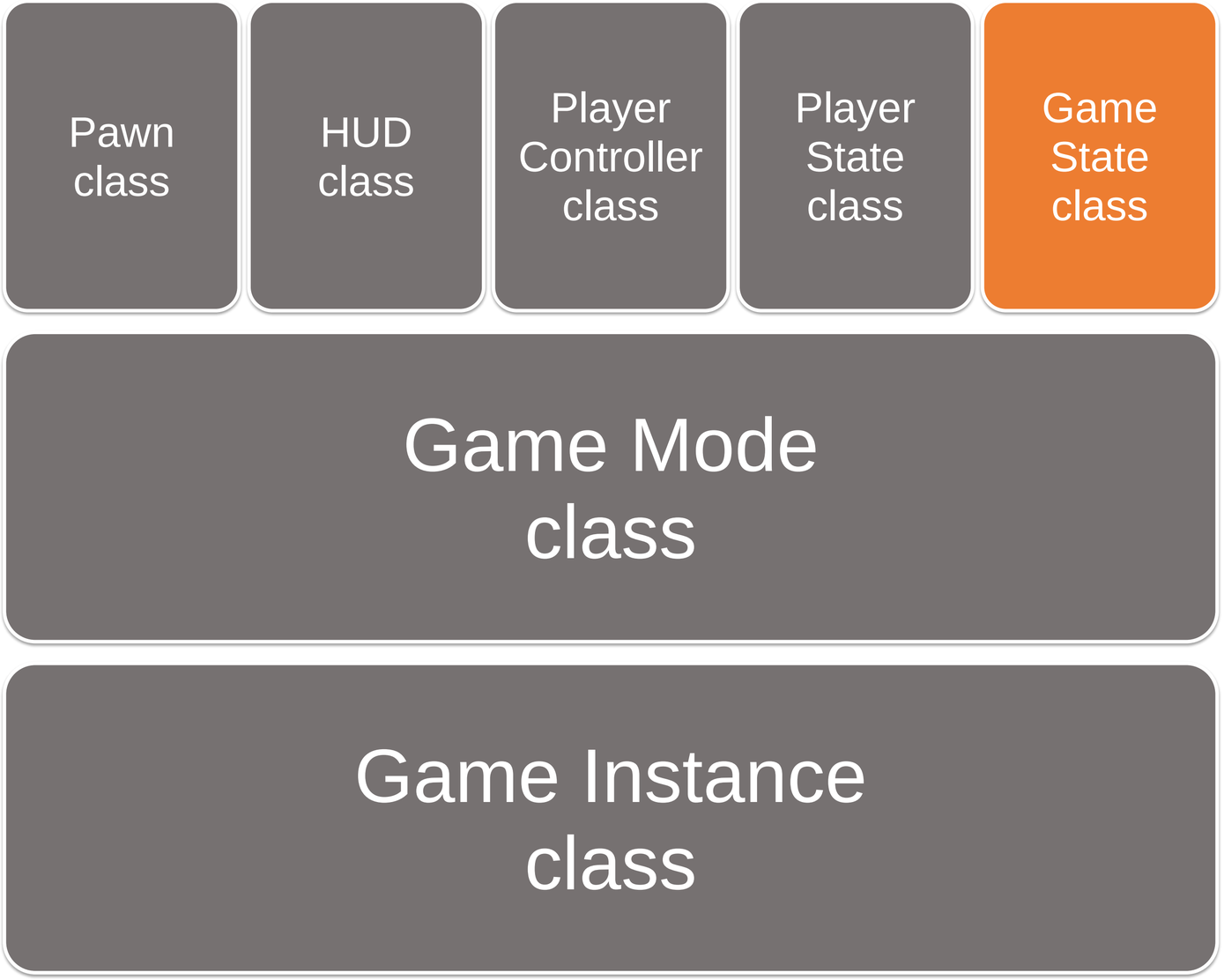
PLAYER STATE CLASS

A Player State is the state of a participant in the game, such as a human player or a bot that is simulating a player.

Non-player AI that exists as part of the game would not have a Player State.

Example data that would be appropriate in a Player State include player name, score, or whether the player is currently carrying the flag in a CTF game.

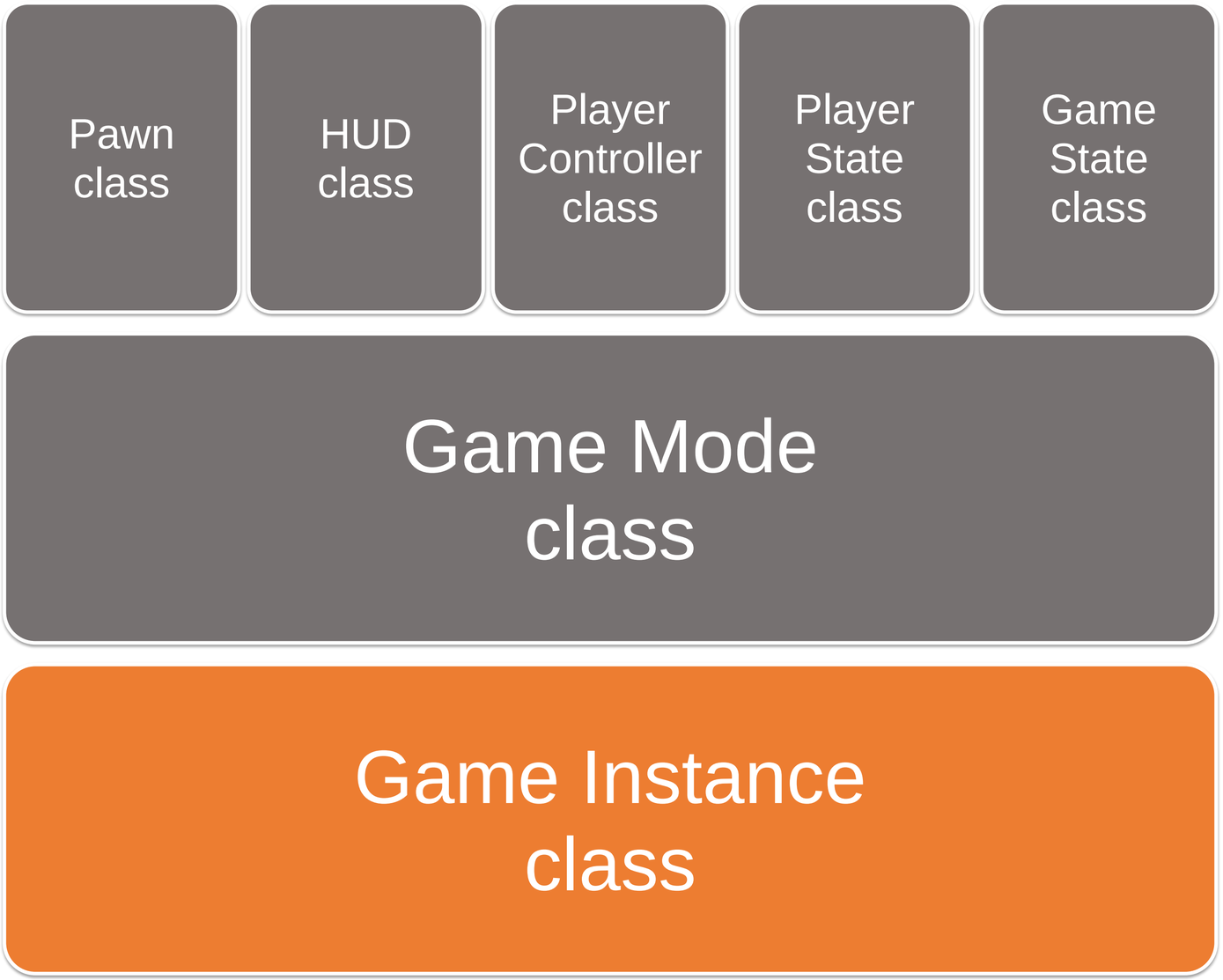
Player States for all players exist on all machines (unlike Player Controllers) and can replicate freely to keep things in sync.



GAME STATE CLASS

The Game State contains the state of the game, which could include things like the list of connected players, the score, where the pieces are in a chess game, or the list of what missions you have completed in an open world game.

The Game State exists on the server and all clients and can replicate freely to keep all machines up to date.



GAME INSTANCE CLASS

The Game Instance is a class whose state withstands the switching of Levels.

Classes like Game Mode or Player Controller are destroyed between Level loads/resets, and data stored in those classes is removed.

Any data that you want to keep between Levels (for example, the player's health and ammo or overall score) should be placed in the Game Instance.

Exercise

- Create a new blank project with starter content or use an existing project.
- In the Content Browser, use the Add New button to add three Game Mode templates of your choice to the project.
- Assign one Game Mode as the default Game Mode under Project Settings. Then create two new Levels for the project and use World Settings properties to assign the other Game Mode templates to each of the Levels.
- Playtest each Level and verify that each Level uses a different Game Mode.