РАЗРАБОТКА DATA DRIVEN АРХИТЕКТУРЫ КЛАССОВ В UNREAL ENGINE 4

Т.В. Монгуш, студент гр. 8В7Б, Ч.Т. Куулар, студент гр. 8И7А, В.А. Коровкин, аспирант гр. А7-39
Томский политехнический университет
E-mail: tvm11@tpu.ru

Введение

В процессе разработки игр реализуется большое количество объектов, которые относятся к различным классам. Если не позаботиться об архитектуре классов, то настанет момент, когда расширение игрового мира новыми объектами станет достаточно сложной задачей. Правильная архитектура экономит много сил и времени на доработку проекта или исправления ошибок, а также повышает производительность работы всей системы в целом.

Unreal Engine 4 – открытый и профессиональный иинструмент для созданияИнтерактивных трехмерных сцен в реальном времени. Внутренняя архитектура классов в Unreal Engine 4 представляет собой легко расширяемую систему под названием Gameplay Framework [1].

Архитектура классов Unreal Engine 4

В UE4 все объекты – это экземпляры классов, которые наследуются от класса UObject. UObject является базовым классом для всех типов объектов в UE.

Объекты, созданные классом UObject не могут находиться на сцене и существуют только в памяти, поэтому их удобно использовать для хранения информации, так как пустой UObject занимает 56 байт и не тратит ресурсы рендера сцены, однако стоит отметить, что UObject можно легко расширить функциями, которые позволят снять ограничения. [2].

Объекты экземпляры классов Actor, которые происходят от базового класса AActor, расширяющий класс Uobject и представляет все игровые объекты, которые могут быть помещены на игровую сцену [3].

Actor используется для обозначения экземпляров классов, которые происходят от AActor, а Objects используется для обозначения экземпляров класса, которые не наследуется от AActor.

Gameplay Framework Classes – это основные классы для представления игроков, союзников и вражеских персонажей, и управление этими персонажами с помощью ИИ или входной логики игрока. Также существуют классы для создания дисплеев и камер для игроков. Игровые классы GameMode, GameState и PlayerState устанавливают правила игрового мира и отслеживают как проходит игровой процесс. Все эти классы создают разные типы актеров, которые могут быть либо помещены на сцену, либо порождены в установленное время.

Pawn расширяет класс Actor и является базовым классом для всех созданий и персонажей, которые контролируются искусственным интеллектом или же непосредственно игроком [1].

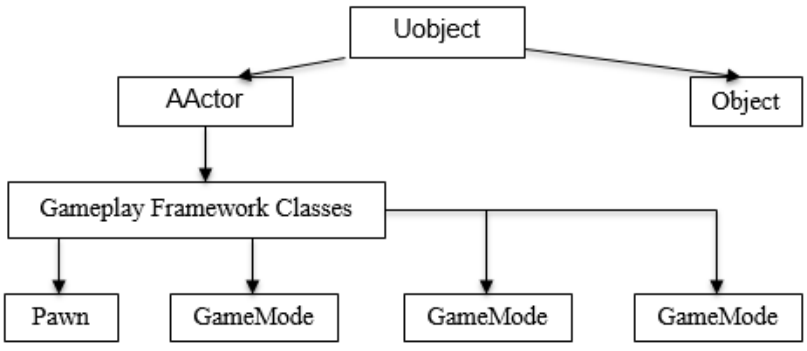

Рис. 1. Архитектура классов в UE4

Разработка интерактивного объекта в Unreal Engine 4

В качестве иллюстрации систем наследования и совместного использования Actor и Object, что позволит в дальнейшем перейти к подходу Data Driven, была реализована интерактивная дверь. Непосредственно реализация интерактивного объекта дверь основана на двух классах: DoorActor и Door.

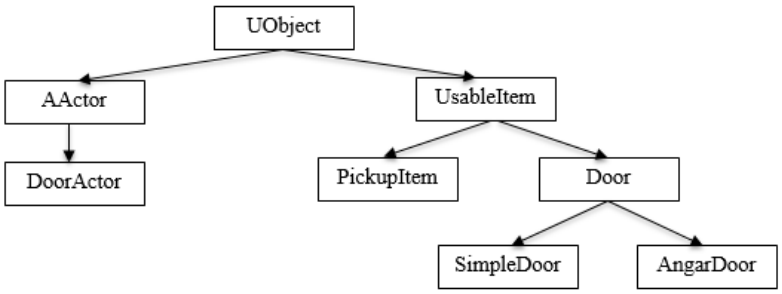

Рис. 2. Архитектура классов реализации двери в UE4

DoorActor унаследован от класса AActor. Именно он отвечает за визуализацию объекта на виртуальной сцене. Door – это класс, хранящий в себе всю информацию о двери. Этот класс унаследован от класса UsableItem, который в свою очередь унаследован от класса Uobject.

Данная архитектура позволяет:

— оптимизировать память и позволяет для хранения и отображения тысяч виртуальных объектов на виртуальной сцене
— удобно сохранять данные с разный момент времени
— удобно редактировать и изменять параметры у объектов (можно использовать различные базы данных).

Рис. 3. Реализованная дверь

Класс DoorActor реализован на C++ и взаимодействует напрямую с классом Door, через получение всех необходимых параметров. А в классе Door основные параметры двери и функции активации событий (например, закрытия и открытия) реализованы на C++, а логика поведения самого события реализована на blueprint коде. Тем самым при помощи класса Door мы можем создавать различные типы дверей, которые бы отличались не только внешне, но и функционально, а класс DoorActor устанавливает свои параметры путем ссылок на данные класса Door. Данный способ намного упрощает процесс расширения разнообразия объектов игрового мира.

Заключение

Unreal Engine является платформой, в которой достаточно удобно контролировать и строить самые разные иерархии классов. Продемонстрированный пример с дверью показывает гибкость и удобство архитектуры Gameplay Framework Unreal Engine 4 для реализации большого количество разнообразных объектов.

Список использованных источников

1. Unreal Architecture [Электронный ресурс] – Режим доступа: https://docs.unrealengine.com/en-US/ProgrammingAndScripting/ProgrammingWithCPP/UnrealArchitecture/index.html, свободный (дата обращения 11.03.21).
2. Расширяем возможности UObject в Unreal Engine 4 [Электронный ресурс] – Режим доступа: https://habr.com/ru/company/pixonic/blog/475622/ , свободный (дата обращения 11.03.21).
3. Иерархия объектов [Электронный ресурс] – Режим доступа: https://sites.google.com/site/unrealdevelopmentkitpractick/home/docs/kopilka-spertyh-statej/unrealscript/ierarhia-obektov, свободный (дата обращения 11.03.21).