M2PW15: Crowd-техники (Реализация толпы)

**ОГЛАВЛЕНИЕ**

[I. Предисловие 2](#_Toc188509012)

[II. Управление 3](#_Toc188509013)

[III. Результаты работы изученных систем 4](#_Toc188509014)

## Предисловие

Для данной практической работы были реализованы Генератор Акторов и, генерируемые им, следующие Акторы-элементы толпы:

1. BP\_Chicken – простой BP-класс, унаследованный от «Pawn»:

Имеет в себе логику «убегания» от Игрока с учётом каких-либо препятствий. Также реализованы расчёт скорости (чем ближе к игроку, тем выше скорость) и поворота головы в сторону направления бега;

1. BP\_Chicken\_cpp – класс, реализованный на языке C++ и имеет ту же логику, что и BP\_Chicken:

Реализован для сравнения по эффективности C++ и Blueprint классов;

1. BP\_Chicken\_BudgetAllocator – BP-класс, унаследованный от «Pawn», с использованием плагина «Animation Budget Allocator» (Распределитель Бюджета Анимации);
2. BP\_Chicken\_DCC\_RVO – класс, подобный BP\_Chicken и унаследованный от «Character», но с использованием «Detour Crowd Controller» (DCC, Менеджер по обходу толпы) и Reciprocal Velocity Obstacles (RVO, Препятствия с обратной скоростью):

По сравнению с BP\_Chicken не имеет самостоятельную логику обходя препятствий и поворота головы;

1. BP\_Chicken\_Sharing – BP-класс, унаследованный от «Actor», с использованием «Animation Sharing Plugin» (Плагин Обмена Анимации):

Не имеет логики поворота головы – не смог полностью совладать с плагином без проявления каких-либо багов;

1. BP\_Chicken\_Niagara\_CPU и BP\_Chicken\_Niagara\_GPU – BP-класс, унаследованный от «Pawn» (хотя можно было бы и от «Actor»), с использованием «Niagara Particle System», запущенном на ресурсах CPU и GPU соответственно:

Не имеет логики поворота головы – с данной системой сложно реализовать плавный поворот головы, так как используется программирование в системе «Niagara» – не изученный мной;

1. BP\_Chicken\_Test – BP-класс без какого-либо «Меша», но с визуализацией «капсулы» коллизии.

## Управление

В проекте реализовано базовое управление персонажем, такие как вперёд-назад (клавиши «W»/«S» или соответствующие “стрелки”), влево-вправо (клавиши «A»/«D» или соответствующие “стрелки”), а также обзор с помощью мыши.

Предусмотрено наличие Паузы для выхода из Игры или смены Локализации. Для вкл./выкл. Паузы задействованы клавиши «Esc», «P» и «Pause».

Для вкл./выкл. Виджета управления Генератором и отображения системы отладки тех или иных задач, задействованы следующие клавиши:

* G и Space (Пробел) – включает/выключает отображение Виджета управления генератором;
* H – запускает информацию о FPS и задержках потоков, а также график этих задержек. По сути, запускает консольную команду (далее к.к.) «Stat Detailed»;
* J – запускает необходимую информацию о работе «Animation Sharing Plugin» (к.к.: «a.Sharing.DebugStates 1/2/3» и «a.Sharing.ToggleVisibility»);
* K – запускает информацию и график работы «Animation Budget Allocator» (к.к.: «a.Budget.Enabled 1», «a.Budget.Debug.Enabled 1» и «Stat AnimationBudgetAllocator»).

## Результаты работы изученных систем

Для понимания эффективности работы представленных реализаций Толпы, было решено выделить следующие данные производительности:

* Усреднённое количество FPS и задержки в потоках CPU и GPU при запуске 1 000 единиц класса-представителя той или иной системы в двух ситуациях:
  1. в отдалении от толпы с обзором на толпу – видна вся толпа, что приводит к нагрузкам изображения анимации;
  2. в центре Генератора толпы – видна часть толпы, но значительная часть Толпы активно реагируют на Персонажа.

Таблица 1 – Данные при 1 000 единиц определённого класса

| Тип | Скрин данных о производительности | |
| --- | --- | --- |
| в отдалении | в центре |
| Blueprint (BP) |  |  |
| C++ |  |  |
| BP\_Budget |  |  |
| BP\_DCC&RVO |  |  |
| BP\_Sharing |  |  |
| BP\_Niagara\_CPU |  |  |
| BP\_Niagara\_GPU |  |  |
| Test |  |  |