2. СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

После того как определены требования к функциональности разрабатываемой системы, ее следует разделить на функциональные блоки. Данный подход облегчит понимание системы, устранит проблемы в архитектуре проекта и позволит создать гибкую и масштабируемую систему для работы в будущем. Но стоит отметить, что деление на функциональные блоки довольно условно в связи с особенностями разрабатываемого проекта.

2.1 Блок приема и обработки зрительной информации

Блок приема и обработки зрительной информации предназначен, как можно понять из названия, прежде всего для имитации зрения персонажей, контролируемых искусственным интеллектом. Из-за относительной сложности построенной системы, обработка данной информации происходит не четко выделенными функциями, а множеством связанных функций, использованных в разных частях искусственного интеллекта. При этом используется относительно простая в реализации логика для приема самой информации.

Основную часть работы по приему зрительной информации делают встроенные функции и компоненты, специально предназначенные и оптимизированные для приема такого рода информации. После добавления необходимого компонента необходимо настроить данный компонент на поиск необходимых персонажей. Далее появится необходимое событие на появление в зоне видимости объектов типа actor. Для корректной работы необходима фильтрация данного массива объектов и их приоритезация.

2.2 Блок приема и обработки звуковой информации

Блок приема и обработки звуковой информации предназначен уже в основном для имитации слуха персонажей. В данном случае, система, отвечающая за слух, имеет больший потенциал для использования. Он позволяет всем необходимым персонажам не только условно слышать окружение, но и косвенно взаимодействовать с другими объектами на карте. Опять же, вся логика, отвечающая за слух искусственного интеллекта, находится не в одном месте, а распределена по разработанной системе.

Логика приема и последующей обработки звуковой информации строится на принципе разделения и тэгирования информации. Так, например, издавая какой-либо звук, можно добавить дополнительную информацию: тэг, громкость,

тип и прочее. Это позволяет передавать не только примитивную информацию о звуке, такую как координаты, но и информацию об источнике и способе создания и прочем.

Основную часть работы выполняют встроенные компоненты и функции, но для более корректной и слаженной работы требуется написание дополнительных алгоритмов. От них будет зависеть, сможет ли персонаж, управляемый искусственным интеллектом получать более подробную информацию об источнике звука. При этом стоит различать звуки, которые может издавать персонаж и звуки, которые слышит игрок при прохождении. Это разные как по целям, так и по функционированию. На основе дополнительной информации можно строить дополнительную логику, совершенствуя интеллект персонажа. Об этом будет более подробно изложено в пункте 2.7 при описании блока коммуникации.

На рисунке 2.1 можно немного детальнее понять, как работают компоненты для блоков, описанных в пункте 2.1 и пункте 2.2. На рисунке показано, как используются описанные компоненты.

Для условного органа слуха создается сфера, она обозначена желтым цветом (цвет не имеет значения и может быть изменен). Всё, что произойдет в сфере, может быть услышано и обработано ботом. При этом существует дополнительная сфера, она не представлена на рисунке в силу большого радиуса и того, что не задействована при разработке. Зеленым показаны границы конуса, используемого как зрение. Все, что попадает в него, также может быть обработано ботом. При это так же важно, что будет именно зрительный контакт, что не будет каких-либо объектов между наблюдателем и наблюдаемым. Для сферы, использованной для слуха это значения не имеет – между наблюдателем и наблюдаемым объектом может быть сколь угодно большое количество объектов.

Разумеется, для правдоподобной системы стоит воспользоваться более сложной логикой. Учитывать при этом тот факт, что звук не может быть услышан за стеной. Но для данного проекта использование встроенного компонента и отсутствие более сложной логики нецелесообразно. Причиной является более сложная система обработки, что повлекло бы дополнительное время разработки. В будущем, при дальнейшей разработке игры планируется добавление такой системы.

Рисунок 2.1 – Органы чувств искусственного интеллекта

2.3 Блок приема и обработки информации о получении урона

Блок приема и обработки информации и получении урона тоже является одним из встроенных способностей искусственного интеллекта в Unreal Engine 4. К сожалению, данное чувство у персонажей в игре не было реализовано в полном объеме по причине свой ситуативности в использовании. Была использована только базовая информация при получении урона: кем был нанесен, с какого направления, величина урона, нормали попадания.

При создании более качественного интеллекта принято писать собственную систему получения урона для повышения эффективности и многофункциональности. Для использования компонента, отвечающего за обработку получения урона, необходимо добавить его к самому персонажу типа actor во внутреннем редакторе персонажа. Далее, как и в случае предыдущих чувств, появится возможность использовать соответствующее событие на получение урона в графе событий персонажа.

2.4 Блок предсказаний

Данный блок отвечает прежде всего за предсказание действий персонажа при движении. Логика проста для понимания – при выходе персонажа из зоны видимости, за которым наблюдает и/или следует искусственный интеллект, делается предсказание. Предсказание представляет собой расчет возможной

точки пребывания наблюдаемого объекта через некоторое время, которое указывается прежде всего в функции напрямую.

Данные, разумеется, можно было взять иным способом, используя данные о местоположении персонажа, запросив их после некоторого промежутка времени. Однако это в некоторых случаях было бы некорректно. Как только персонаж скрывается за условной стеной, может произойти изменение вектора направления движения, чего не может знать персонаж, управляемый искусственным интеллектом. Уже говорилось ранее, что стоит ограничивать количество таких запросов. Основной причиной желательной минимизации запросов таких данных служит возможная неправдоподобность действий интеллекта при их использовании, что может не понравится пользователю. Добавление такого компонента также не составляет труда – из того же списка доступных при добавлении чувств, необходимо лишь выбрать данный компонент. После добавления появится соответствующий обработчик события. Логика, созданная при помощи данного компонента совместно с использованием blackboard для хранения данных, позволяет добавить интеллекту некоторое подобие памяти.

2.5 Блок принятия решений

Блок принятия решений является одним из основных в системе. Он обрабатывает всю информацию, полученную от так называемых органов чувств персонажа. В некоторых случаях используется дополнительное обращение за данными напрямую к объектам окружения для минимизации хранимой информации. В некоторых случаях хранение данных нецелесообразна, так как нет определенных событий для записи их в хранилище. Данный блок по своей сути объединяет всю логику обработки органов чувств и после использования блока приоритезации дает разрешение на выполнение необходимой задачи.

Описываемый блок является максимально распределенным и опять же не может быть описан в группе функций, предназначенных только для реализации этого блока. Для корректной работы он использует не только блок приоритезации, но и блок обработки и выполнения принятых решений. Причинами тому являются, в некоторых исключительных ситуациях, невозможность выполнения поставленных искусственному интеллекту задач. Вспомогательный блок обработки и выполнения не совсем корректно считать блоком обработки ошибок, однако они схожи по выполняемым функциям.

Тут же стоит упомянуть и блок коммуникации, который тоже влияет на принятие решений. При коммуникации между собой, противники получают

дополнительную информацию. В исключительных случаях, требуется непосредственный опрос окружения на предмет необходимой информации. Одним из примеров является опрос команды для получения информации о возможном местонахождении игрока.

2.6 Блок выполнения задач

Блок выполнения задач в свою очередь выполняет задачи, поставленные блоком принятия решений. Задачи выбираются не напрямую, а выставлением необходимых переменных в необходимые значения. Например, для того, чтобы начать или продолжить патрулирование, персонаж не должен видеть игрока, или иной объект, угрожающий ему. Это одно из условий выбора данной задачи. При невозможности выполнения какой-либо задачи необходимо установить соответствующие значения переменных и пересмотреть необходимость выполнять поставленную задачу. Опять в пример можно привести патрулирование. При нахождении игрока, что является целью патрулирования, противник, контролируемый искусственным интеллектом, перестает патрулировать и идет выбор новой приоритетной задачи. Примерами таких задач может быть попытка убежать в укрытие или попытка атаки игрового персонажа, управляемого пользователем. В силу сложности реализации четко разделенных блоков в искусственном интеллекте, блок выполнения задач тоже распределен и не может быть выделен как отдельная группа функций, созданных исключительно для выбранных целей.

2.7 Блок коммуникации между объектами

Блок коммуникации прежде всего является частью обработки звуковой информации. Как уже было кратко описано выше, в описании блока приема и обработки звуковой информации, блок коммуникации использует дополнительную информацию о звуках вокруг. Так как информация о звуке тэгирована, а значит содержит какое-либо закодированное сообщение, то необходима также и особая обработка такой информации. Для этих целей был выделен отдельный блок обработки.

Основной информацией, которую может получить искусственный интеллект, является причина появления звука. Причина включает в себя как источник, так и неоговоренный приоритет обработки информации. Например, при выстреле игрок издает звук, который может уловить противник. Если он его уловит, он может получить примерное местоположение игрока и пропустит

патрулирование, решив исследовать область выстрела. Причиной приостановки задачи патрулирования может быть также условное общение между персонажами, управляемыми искусственным интеллектом. В данном случае персонажи издают звуки с особым тэгом с заданными интервалом. Также есть некоторый приоритет у разных видов персонажей. Так, например, летающий искусственный интеллект, дрон, может посылать информацию персонажам, но при получении одновременно информации от другого идентичного персонажа и дрона, приоритетнее будет информация персонажа, а не дрона. Это связано прежде всего с едва заметной попыткой сделать общение более живым. Для более интеллектуальных противников приоритетной информацией будет информация, полученная от более интеллектуальных противников. Но за неимением такой, персонаж выберет информацию дрона и выполнит соответствующую задачу.

2.8 Блок приоритезации данных органов чувств

Блок приоритезации является относительно небольшим в разрабатываемой системе, но также он является значимой ее частью. Он тесно связан с блоком принятия решений и в некоторых случаях они могут выполнять максимально схожие по целям задачи. Но их разграничение необходимо прежде всего для обработки исключительных ситуаций. Он так же связан с блоком коммуникации и блоком обработки и выполнения принятых решений, но связан косвенно, через блок принятия решений. Ярко выраженным примером приоритезации является отдельная ветвь в дереве поведений. Данная ветвь выполняет приоритетную задачу осмотра локации после выстрелов, совершенным игроком, или иных действий, которые персонаж может услышать. Она выполняется после приема информации от блока коммуникации, обработки в блоке принятия решений и опроса возможных персонажей в радиусе. Остальная часть приоритезации сильно распределена по системе в силу невозможности выделения ее в отдельный блок или подобную функцию. Без данного блока было бы невозможно существование задач, требующих незамедлительной обработки. Приоритеты в каждого объекта на карте или, вернее сказать, уровне, заданы неявно. Они привязаны непосредственно к типу объекта, отдельной переменной или иного способа предусмотрено не было по причине отказа от дополнительной загруженности переменными.