Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ***Робототехника и комплексная автоматизация***

КАФЕДРА ***Системы автоматизированного проектирования (РК-6)***

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Студент Колыхалов Дмитрий Витальевич
фамилия, имя, отчество

Группа ***РК6-84Б***

Тип практики ***Преддипломная***

Название предприятия ***НИИ АПП МГТУ им. Н.Э. Баумана***

Студент _____ **Колыхалов Д. В.**
подпись, дата *фамилия, и.о.*

Руководитель практики
от кафедры _____ Оглоблин Д. И.
подпись, дата *фамилия, и.о.*

Оценка _____

2025 г.

«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой *РК6*
_____ __ *А.П. Карпенко* _
« ___ » _____ 2025 г.

З А Д А Н И Е
на прохождение производственной практики
_____***Преддипломная***_____
Тип практики

Студент

__Колыхалов Дмитрий Витальевич___ __***4***__ курса группы _***РК6-84Б***_
Фамилия Имя Отчество № курса индекс группы

в период с _***01 мая 2025***__ г. по ***27 мая 2025*** г.

Предприятие: «НИИ Автоматизации Производственных Процессов МГТУ им. Н.Э. Баумана»
Подразделение:

_____
(отдел/сектор/цех)

Руководитель практики от предприятия (наставник):
Витюков Фёдор Андреевич
(Фамилия Имя Отчество полностью, должность)

Руководитель практики от кафедры:
Оглоблин Дмитрий Игоревич
(Фамилия Имя Отчество полностью, должность)

Задание:
1. *Разработка персонажей окружения.*
2. *Разработка боевой системы.*

Дата выдачи задания *1 мая 2025* г.

Руководитель практики от предприятия _____/ Витюков Ф.А. /

Руководитель практики от кафедры _____/ Оглоблин Д.И. /

Студент _____/ Колыхалов Д. В. /

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Компьютерная графика (машинная графика) — это система методов, алгоритмов, программных и аппаратных средств для ввода, обработки и отображения графической информации, а также для преобразования данных в графическую форму.

Компьютерная графика позволяет создавать, изменять, анализировать, стирать изображения, а также работать с цветом и яркостью всего изображения и отдельных его фрагментов, реализовывать на экране дисплея движущееся цветное изображение.

Основные области применения компьютерной графики:

- конструирование и дизайн различных объектов и систем;
- интернет-дизайн;
- обработка и монтаж цифровых изображений;
- подготовка полиграфической и картографической продукции;
- компьютерная анимация;
- создание спецэффектов для кино и телевидения;
- виртуальные объекты и пространства (компьютерные игры, тренажёры).

В ходе прохождения практики требуется выполнить следующие цели:

1. Разработка персонажей окружения;
2. Разработка боевой системы.

Для достижения данной цели следует разобраться со следующими понятиями, такими как Unreal Engine.

Unreal Engine — это игровой движок от Epic Games, одной из крупнейших американских компаний по разработке игр и программного обеспечения. И хотя инструмент предназначен в первую очередь для создания видеоигр, он подходит и для производства неигровых проектов в области архитектуры, строительства, автомобильной промышленности, медицины, кинематографа, анимации и других сфер. Редактор движка Unreal Editor создан по принципу «что пользователь видит, то и получит». Это

означает, что итоговый результат не будет отличаться от его изображения в 3D-вьюпорте. К тому же редактор очень удобен для использования: все ассеты (модели, источники освещения, визуальные эффекты и так далее) можно сразу разместить в сцене, перетащив из папок. В целом Unreal Editor можно назвать комплексной системой, состоящей из многочисленных редакторов, которая направлена на то, чтобы сделать процесс разработки максимально цельным.

Для реализации цели «Разработка персонажей окружения» ее следует разбить на следующие задачи:

1. Создание ИИ для контроля действий персонажа окружения;
2. Настройка анимаций персонажей.

Для реализации цели «Разработка боевой системы» ее следует разбить на следующие задачи:

1. Настройка анимаций персонажа окружения;
2. Захват цели;
3. Комбинация атак;
4. Система здоровья;
5. Система уклонений;
6. Вражеский персонаж окружения;
7. Главный вражеский персонаж окружения.

1. Разработка персонажей окружения

1.1. Цель работы

Целью данной работы является разработка персонажа окружения на языке C++.

1.2. Задание

Разработать персонажа окружения на языке программирования C++ с использованием функционала игрового движка Ureal Engine, результатом выполнения которого является преследование главного персонажа после попадания в зону видимости персонажа окружения.

1.3. Задачи

Для выполнения целей необходимо выполнить следующие задачи:

1. Создание ИИ для контроля действий персонажа окружения;
2. Настройка анимаций персонажа.

1.4. Реализация

Персонаж окружения – неигровой персонаж, который служит важным средством создания атмосферы в сцене. В Unreal Engine логика неигрового персонажа осуществляется за счет Behavior Tree – дерево поведения, компонентом которого служит Blackboard. Blackboard – это ресурс, который хранит все данные, необходимые для конкретного контроллера AI, на который будет ссылаться дерево поведения.

Для реализации NPC было выбрано три его основных поведения:

1. Find Random Location – данное поведение реализуется случайную ходьбу по карте неигрового персонажа.
2. Find Player Location – поведение, которое реализует нахождение главного персонажа на сцене. Персонаж будет являться найденным если NPC его увидел в определенном поле своего зрения.
3. Chasing Player – поведение реализующее преследование главного персонажа. Преследование продолжается до тех пор, по NPC видит главного персонажа в своей определенной области.

Разработанное дерево поведения NPC представлено на рисунке 1.

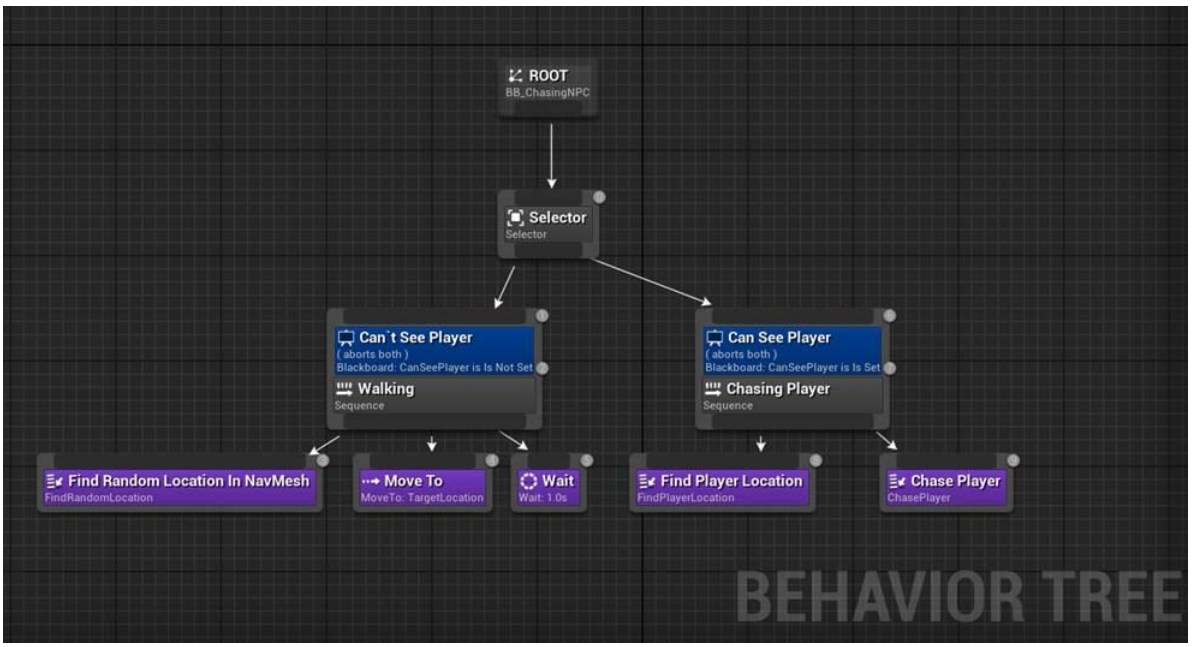

Рис. 1. Дерево поведения персонажа окружения

В ходе реализации персонажа окружения были созданы необходимые классы на языке программирования C++ с использованием технологии Blueprints для контроля необходимых анимаций персонажа.

1. ChasingNPC – класс персонажа окружения;
2. FindRandomLocation – класс поиска случайно локации;
3. FindPlayerLocation – класс поиска главного персонажа;
4. ChasePlayer – класс преследования главного персонажа;
5. ChasingNPC_AIController – класс служащий для контроля действий персонажа окружения.

Блок-схема персонажа окружения представлена на рисунке 2.

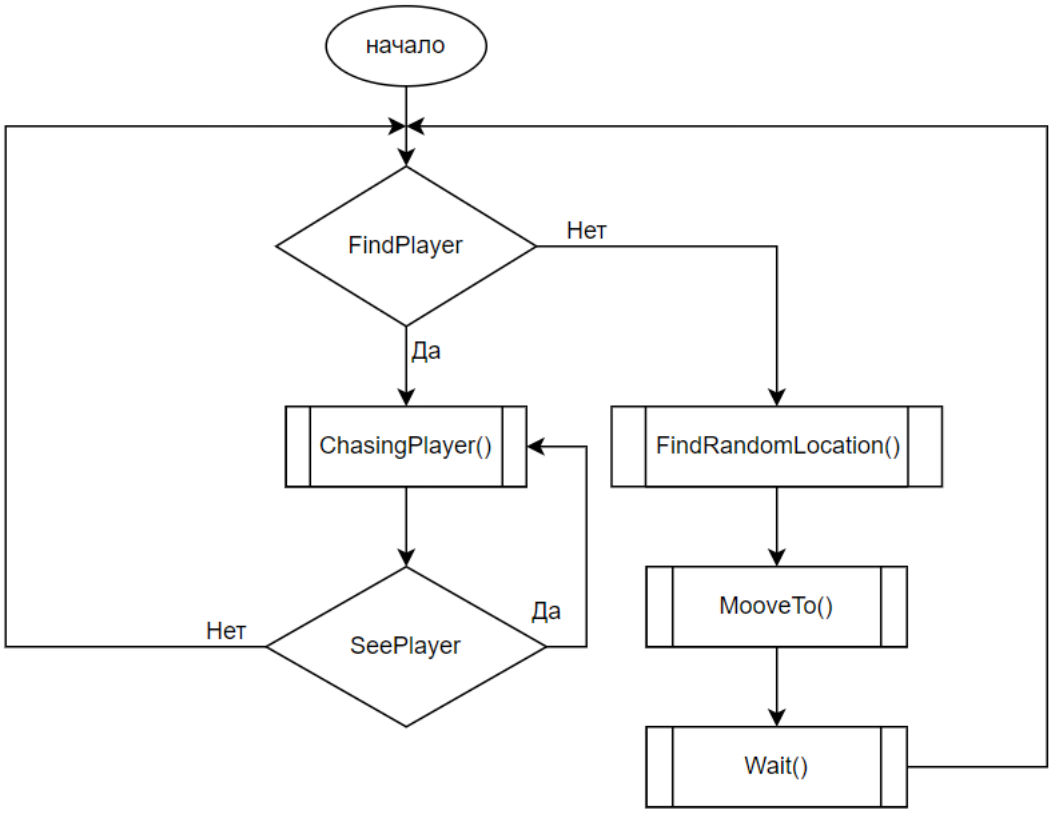

Рис. 2. Блок-схема персонажа окружения

Листинг 1. Реализация функции поиска случайно локации

```
EBTNodeResult::Type
UBTTask_FindRandomLocation::ExecuteTask(UBehaviorTreeComponent&
OwnerComp, uint8* NodeMemory)
{
	// get AI controller and it`s NPC
	if (auto* const cont =
Cast<AChasingNPC_AIController>(OwnerComp.GetAIOwner()))
	{
		if (auto* const npc = cont->GetPawn())
		{
			// obtain npc location to use as an origin
			auto const Origin = npc->GetActorLocation();

			// get the navigation system and generate a random
location
			if (auto* const NavSys =
UNavigationSystemV1::GetCurrent(GetWorld()))
			{
				FNavLocation Loc;
				if (NavSys-
>GetRandomPointInNavigableRadius(Origin, SearchRadius, Loc))
				{
					OwnerComp.GetBlackboardComponent()-
>SetValueAsVector(GetSelectedBlackboardKey(), Loc.Location);
				}

				// finish with success
				FinishLatentTask(OwnerComp,
EBTNodeResult::Succeeded);
				return EBTNodeResult::Succeeded;
			}
		}
	}

	return EBTNodeResult::Failed;
}
```

Листинг 2. Функция, реализующая поиск главного персонажа

```
EBTNodeResult::Type
UBTTask_FindPlayerLocation::ExecuteTask(UBehaviorTreeComponent&
OwnerComp, uint8* NodeMemory)
{
	// get player character
	if (auto* const Player =
UGameplayStatics::GetPlayerCharacter(GetWorld(), 0))
	{
		// get player location to use as an origin
		auto const PlayerLocation = Player->GetActorLocation();
		if (SearchRandom)
		{
			FNavLocation Loc;

			// get the navigation system and generate a random
location near the player
			if (auto* const NavSys =
UNavigationSystemV1::GetCurrent(GetWorld()))
			{
				// try to get a random location near the player
				if (NavSys-
>GetRandomPointInNavigableRadius(PlayerLocation, SearchRadius, Loc))
				{
					OwnerComp.GetBlackboardComponent()-
>SetValueAsVector(GetSelectedBlackboardKey(), Loc.Location);
					FinishLatentTask(OwnerComp,
EBTNodeResult::Succeeded);
					return EBTNodeResult::Succeeded;
				}
			}
		}
		else
		{
			OwnerComp.GetBlackboardComponent()-
>SetValueAsVector(GetSelectedBlackboardKey(), PlayerLocation);
			FinishLatentTask(OwnerComp,
EBTNodeResult::Succeeded);
			return EBTNodeResult::Succeeded;
		}
	}
	return EBTNodeResult::Failed;
}
```

Листинг 3. Функция, реализующая преследование главного персонажа

```
EBTNodeResult::Type
UBTTask_ChasePlayer::ExecuteTask(UBehaviorTreeComponent&
OwnerComp, uint8* NodeMemory)
{
      // get target location from blackboard via the NPC`s
controller
      if (auto* const cont =
Cast<AChasingNPC_AIController>(OwnerComp.GetAIOwner()))
      {
            auto const PlayerLocation =
OwnerComp.GetBlackboardComponent()-
>GetValueAsVector(GetSelectedBlackboardKey());

            // move to the player`s location
            UAIBlueprintHelperLibrary::SimpleMoveToLocation(cont,
PlayerLocation);

            //finish with success
            FinishLatentTask(OwnerComp, EBTNodeResult::Succeeded);
            return EBTNodeResult::Succeeded;
      }
      return EBTNodeResult::Failed;
}
```

2. Разработка боевой системы

2.1. Цель работы

Целью данной лабораторной работы разработка боевой системы между персонажем окружения и главным персонажем.

2.2. Задание

Разработать боевую системы между главным персонажем и персонажем окружения.

2.3. Задачи

Для выполнения работы следует реализовать следующие задачи:

1. Настройка анимаций персонажа окружения;
2. Захват цели;

3. Комбинация атак;
4. Система здоровья;
5. Система уклонений;
6. Вражеский персонаж окружения;
7. Главный персонаж окружения.

2.4. Реализация

2.4.1. Настройка анимаций перемещения

Для реализации анимаций перемещения была использована технология Blend Space, которая позволяет плавно смешивать анимации на основе определенных параметров, таких как скорости, направления движения, угла атаки и пр. Данная технология позволяет создавать естественные переходы между анимациями, адаптируя их к динамике процесса. При создании Blend Space используются следующие параметры: ось X – направление, где -1 это движение влево, 1 – вправо и ось У – скорость. Полученный граф анимаций перемещения персонажа представлен на рисунке 3.

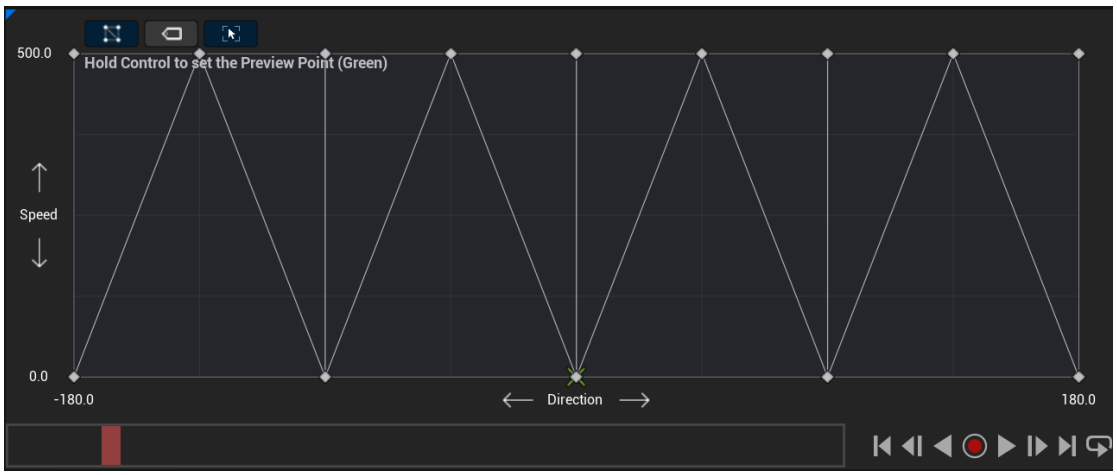

Рис. 3. Граф анимаций перемещения персонажа

После создания Blend Space контроля анимаций был разработан Animation Blueprint, который позволяет использовать набор анимаций из Blend Space на персонаже. Так как полученный набор анимаций использует параметр направления, то в Animation Blueprint должен находиться блок, позволяющий получить направление движения персонажа. Для большей реалистичности передвижения персонажа следует использовать Camera Lag –

параметр, который задерживает перемещение камеры в зависимости от направления движения персонажа на определенное время, что позволяет добиться перемещение персонажа до перемещения камеры. Разработанный Animation Blueprint представлен на рисунке 4.

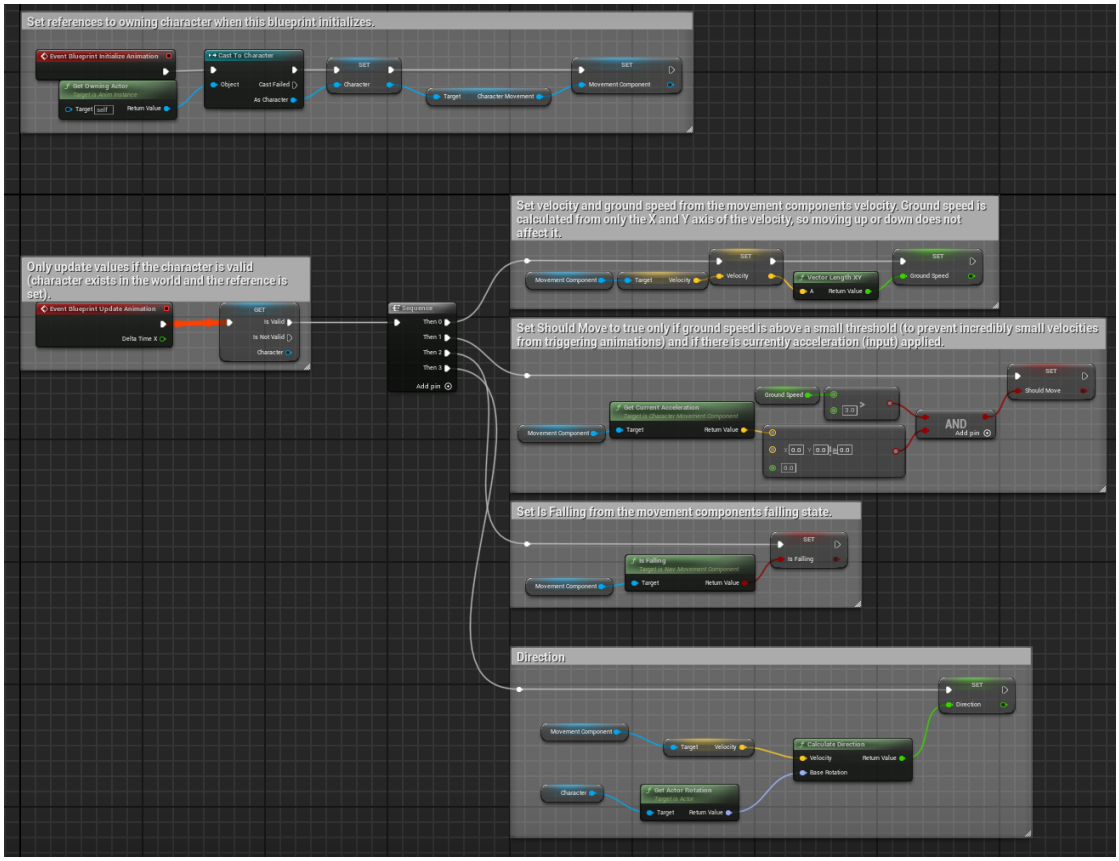

Рис. 4. Animation blueprint главного персонажа

2.4.2. Захват цели

При создании боевой системы важной механикой является захват цели. Она позволяет пользователю автоматически фиксировать внимание на конкретном объекте сцены, также используется для упрощенного управления, повышения точности, что добавляет больше динамический возможностей. Так как захват цели является дополнительной механикой, то для него было разработано отдельное входное действие, которое было добавлено в IMC_Defaults – набор всех входных действий, в котором задается клавиша, при нажатии которой будет производиться захват цели.

Для реализации функциональной части захвата цели был создан Blueprint Class Actor Component – BPC_Combat, что будет являться частью

основного класса персонажа Actor и реализовывать дополнительный функционал, отвечающий за боевые механики персонажа. Ключевой особенностью захвата цели является то, что направление зрения персонажа должно быть на цели, на которую происходит захват. Данная особенность должна не только удерживать зрение персонажа на цели, но и сопутствовать ограничениями при движениях, что даст большую реалистичность возможности захвата цели. Для реализации был создан визуальный интерфейс посредством Widget Blueprint – TargetLock. Данный интерфейс отображается поверх объекта захвата цели, тем самым показывая, что захват цели произведен. Для отображения визуального интерфейса захвата цели был создан класс Actor – BP_TagetLockWidget. Данный класс является дополнительным объектом в сцене. Разработанный блок захвата цели представлен на рисунке 5.

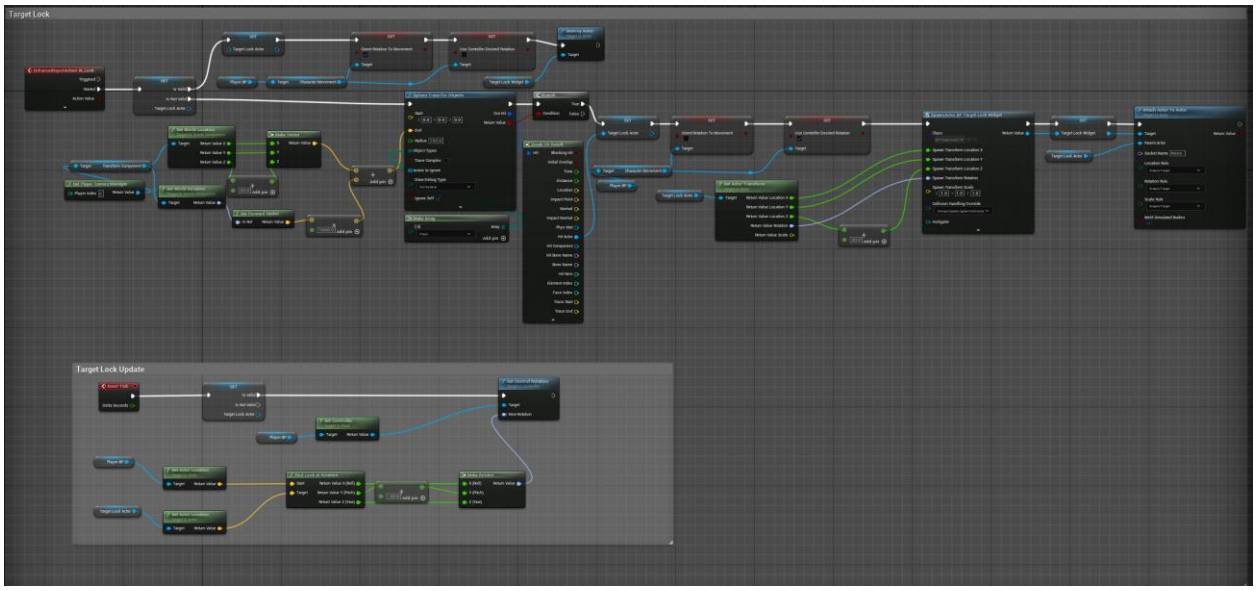

Рис. 5. Блок захвата цели

2.4.3. Комбинация атак

Основной частью боевой системы является комбинация атак. Это последовательность действий, которые пользователь выполняет для достижения определенных целей прогресса в сцене.

Для реализации комбинации атак было создано дополнительное входное действия аналогичное захвату цели, но различное в клавише вызова этого

действия. В уже разработанном ранее классе BPC_Combat была создана новая функция действия, отвечающая за атакующие действия персонажа. Важной особенностью при создании боевой системы атак является полное проигрывание анимации атаки, т е невозможности ее прерывания последующим нажатием на клавишу атаки. Чтобы достичь большую реалистичность и разнообразие атак, был создан Animation Montage – монтаж определенных анимаций, в котором возможно покадровое редактирование анимации и добавление специальных маркеров, которые отвечают за определенные действия во время проигрывания анимации. Ключевой особенность анимационного монтажа является разделение анимации атаки на несколько различных, что дает возможность реализации различных комбинаций атак, таких как одиночная, двойная и пр. Разработанный блок анимаций комбинаций атак представлен на рисунке 6.

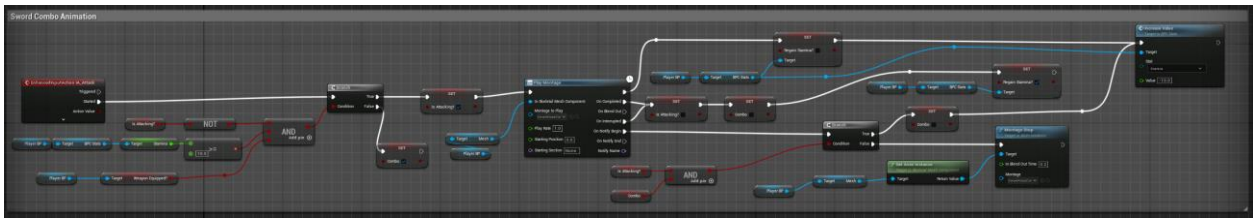

Рис. 6. Блок анимаций комбинаций атак

При реализации в боевой системе комбинаций атак помимо самих анимаций должно присутствовать нанесение урона. Для реализации оповещения нанесенного урона был разработан класс Anim Notify State – BP_Notify_Damage. Логика работы данного класса заключается в оповещении наносимого урона во время начала и конца проигрывания анимации атаки. Для возможности получать данные оповещения при атаках персонажа, в классе BPC_Combat был разработан блок Damage Trace, отвечающий за связывание получения оповещения о наносимом уроне с атаками персонажа. Для возможности наносить урон был создан специальный разъем в скелете персонажа, который является местом, где персонаж держит оружие. В классе персонажа в компоненте его сетки была создана дополнительная сетка, являющаяся мечем. В самом мече были созданы маркеры начала и конца

оружия, при прохождении прямой, которую они образуют, через сетку другого объекта будет создаваться наносимый урон. Разработанный блок отслеживания движения меча и оповещения наносимого урона представлен на рисунке 7.

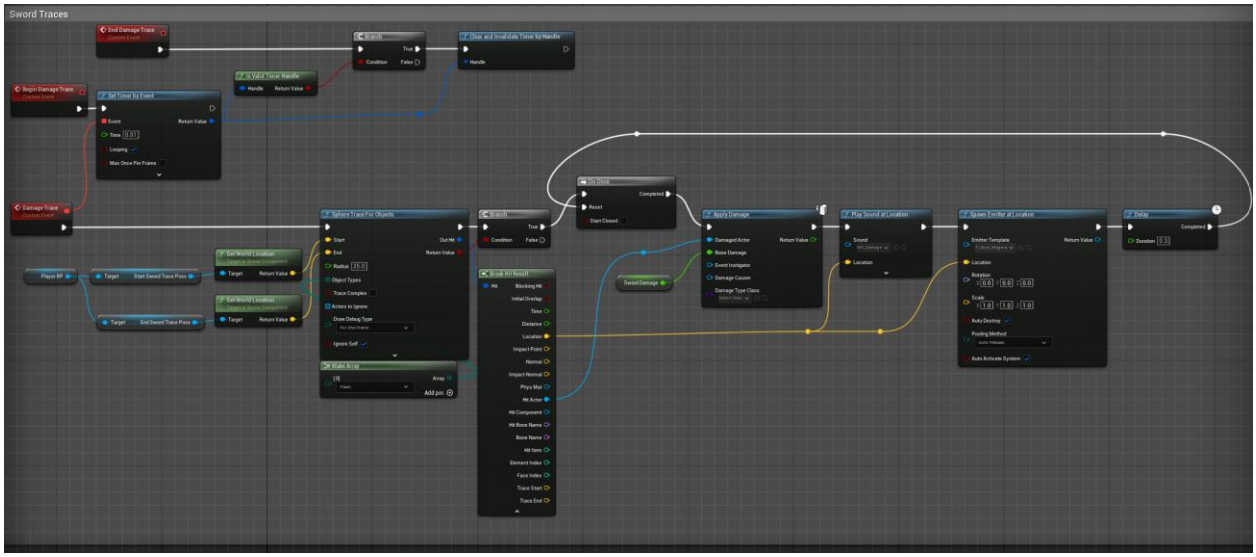

Рис. 7. Блок отслеживания меча и оповещения наносимого урона

2.4.4. Система здоровья

Система здоровья – один из ключевых механизмов, определяющий «живучесть» персонажа или объекта. Она влияет на баланс сложности, прогрессии и развития, тактические решения и сюжетные взаимодействия. Система здоровья имеет разные типы:

1. Статическое здоровье – фиксированный запас здоровья, который не восстанавливается автоматически.

2. Регенерация – здоровье восстанавливается со временем, в укрытие или при использовании специальных предметов, повышающих текущий уровень здоровья.

3. Сегментированное здоровье – тип, при котором здоровье делится на части, восстановление которых происходит по-отдельности.

4. Броня и здоровье – вариант контроля здоровья, при котором изначально расходуется запас брони, затем здоровья.

5. Системы с состояниями – система, при которой повреждение влияют на дальнейшую функциональность (сломанная нога, что влечет за собой медленное перемещение по сцене).

Во время реализации системы здоровья был создан класс Actor Component – BPC_Stats, где ключевыми переменными являются здоровье персонажа и его максимальное здоровье. В самом классе была реализована функция Increase Value, которая обновляет значения статистик персонажа. Для связи и контроля статистик персонажа был создан набор данных Enumerations – E_Stats, который позволяет задать набор именованных констант для работы с логикой проекта. Данный набор используется для замены чисел на читаемые метки. Разработанная функция представлена на рисунке 8.

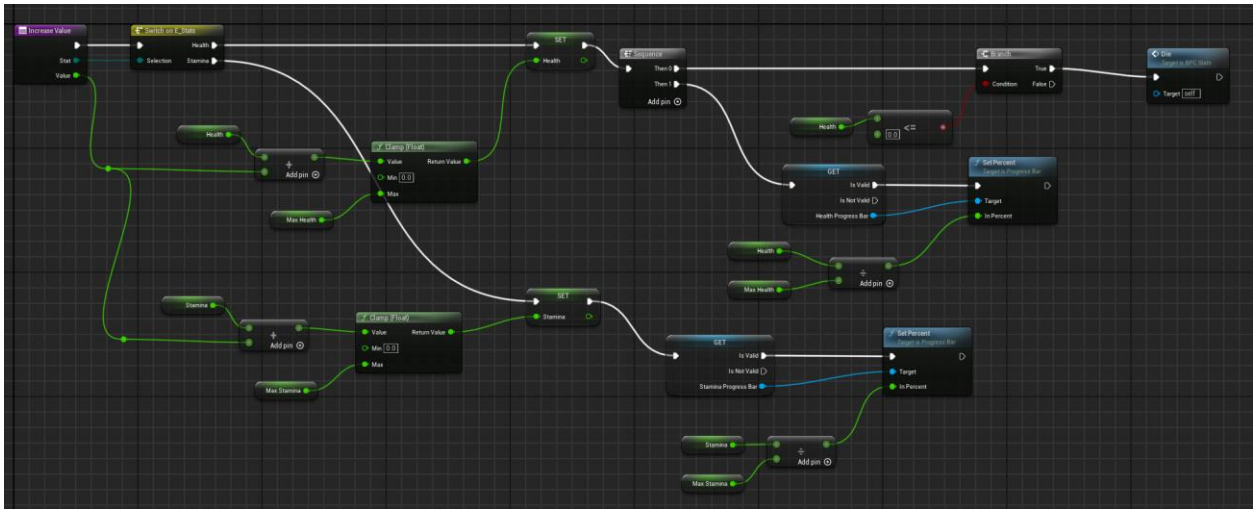

Рис. 8. Функция обновления статистик персонажа

Важной особенностью в системе здоровья является ее реакция на полученный персонажем урон. Для реализации этой функция в классе персонажа был создан блок Damage, который осуществляет вычитание полученного урона из текущего здоровья после оповещения, которое было разработано в комбинации атак. Для большей реалистичности получения урона создан монтаж анимации HitReact и ограничена возможность перемещения персонажа при его получении. Разработанный блок представлен на рисунке 9.

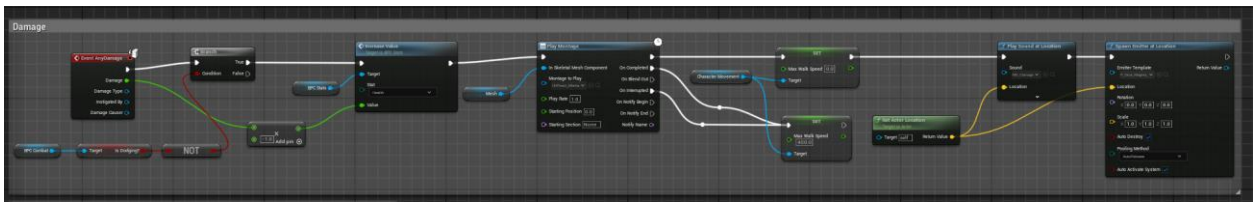

Рис. 9. Блок контроля здоровья при нанесении урона

Ключевой механикой при реализации системы здоровья является смерть персонажа, получаемая при окончании запаса здоровья. Для ее реализации была создана функция Die в классе персонажа, в которой происходит ограничение передвижение персонажа и невозможность его управления. Полученная функция представлена на рисунке 10.

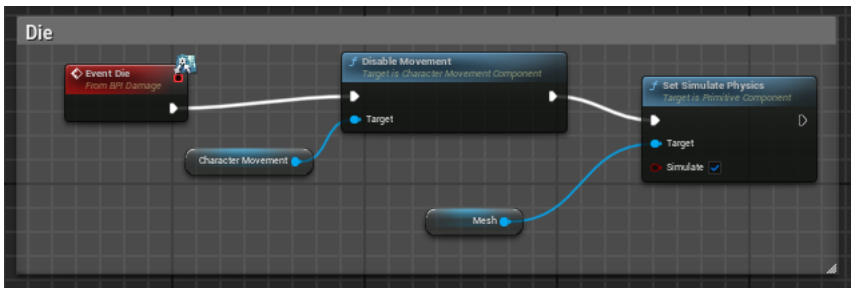

Рис. 10. Функция смерти персонажа

При разработке системы здоровья персонажа был создан виджет WB_PlayerHealth и пользовательский интерфейс WB_HUD для визуализации здоровья. Для отображения состояния здоровья в сцене в классе персонажа были созданы блоки, представленные на рисунке 11.

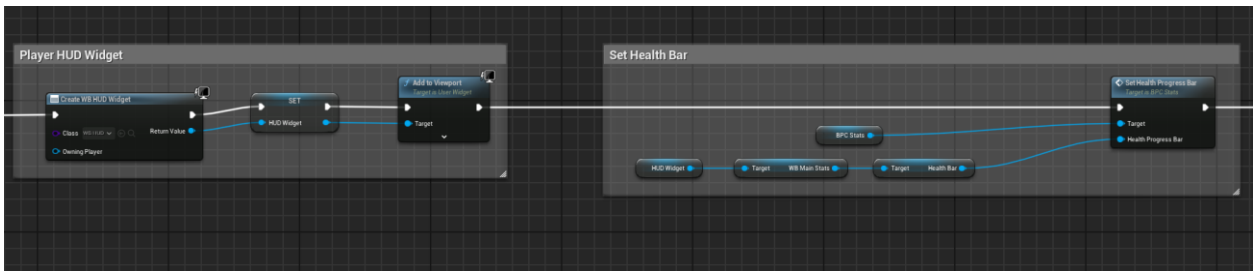

Рис. 11. Блоки отображения состояния здоровья персонажа

Важной особенностью системы здоровья является ее восстановление. Для ее реализации были созданы анимационный монтаж, входное действие, при нажатии клавиши которого будет происходить действие восстановления здоровья. Разработанный блок механики восстановления здоровья представлен на рисунке 12.

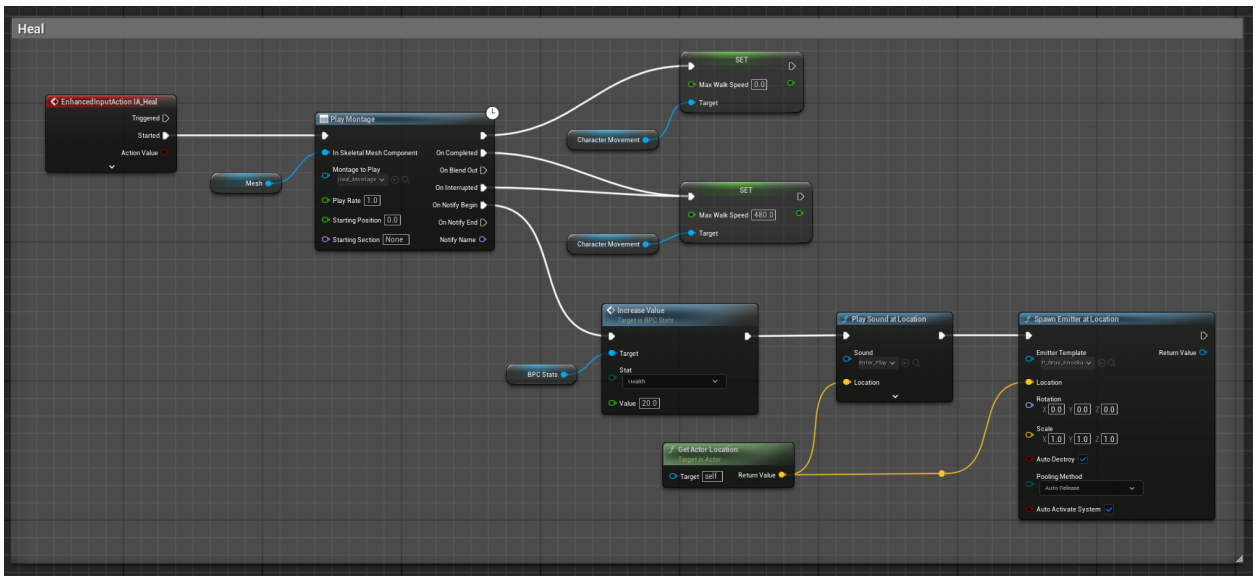

Рис. 12. Блок восстановления здоровья персонажа

2.4.5. Система уклонения

Уклонение – это механика, позволяющая персонажу избегать получение урона, контролировать позиционирование в бою или создавать возможности для атаки. Данная механика используется для большей реалистичности в сцене, повышения динамики боя, снижения зависимости от здоровья и неким балансом между риском и победой.

Механика уклонения может быть представлена разными типами:

1. Активное уклонение – вид уклонения, требующий нажатия специальной клавиши для совершения действия.
2. Пассивное уклонение – имеет автоматический характер при выполнении определенных условий.
3. Контрудары – вид уклонения, при котором происходит уклонение с последующей атакой.
4. Использование окружения – укрытие за объектами сцены.

Для реализации системы уклонения были созданы монтаж анимаций, входное действие при нажатии клавиши. В классе BPC_Combat разработан блок – Dodging, в котором при нажатии определенной клавиши происходит проигрывание анимации уклонения в соответствующем направлении движения персонажа, которое определяется функцией MovementDirection в

классе персонажа и набором данных E_MovementDirection. Разработанный блок уклонения представлен на рисунке 13.

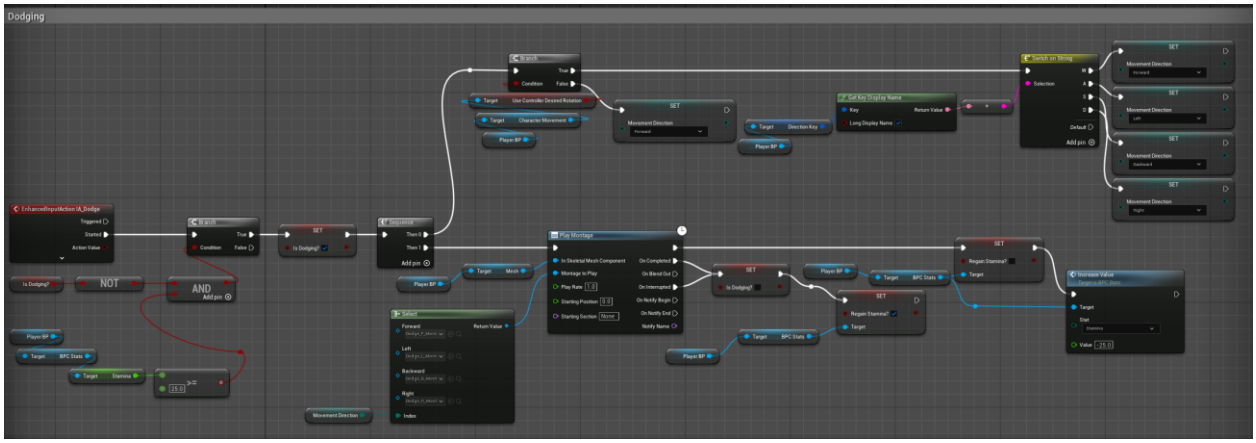

Рис. 13. Блок системы уклонения персонажа

2.4.6. Вражеский персонаж окружения

При разработке боевой системы необходимым условием ее реализации является создание вражеского персонажа окружения.

Для реализации вражеского персонажа окружения были созданы:

1. BD_AI – blackboard доска, служащая для хранения и управления данными, которые используются искусственным интеллектом в рамках системы Behavior Tree (дерево поведения.)
2. BT_Enemy – дерево поведения вражеского персонажа окружения, которое будет содержать в себе возможные действия персонажа в зависимости от определенных ситуаций, происходящих в сцене.
3. Класс AIController – BP_AI_Enemy – класс вражеского персонажа окружения, который содержит в себе его дерево поведения и данные для реализации его реализации.
4. Класс Character – BP_Enemy – основной класс вражеского персонажа окружения.

В BD_AI были созданы следующие ключи для реализации поведения вражеского персонажа окружения:

1. seeingTarget? – ключ, отвечающий за видение цели атаки или преследования.

2. Target – вражеский объект, который необходимо атаковать.

В дереве поведения вражеского персонажа окружения разработаны следующие задания, реализующие определенные его действия:

1. Task_ChaseTarget – функция, реализующая преследование заданной цели.

2. Task_SwordAttack – функция, служащая для атаки заданной цели вражеским переснажем.

Разработанное дерево поведения вражеского персонажа окружения представлено на рисунке 14.

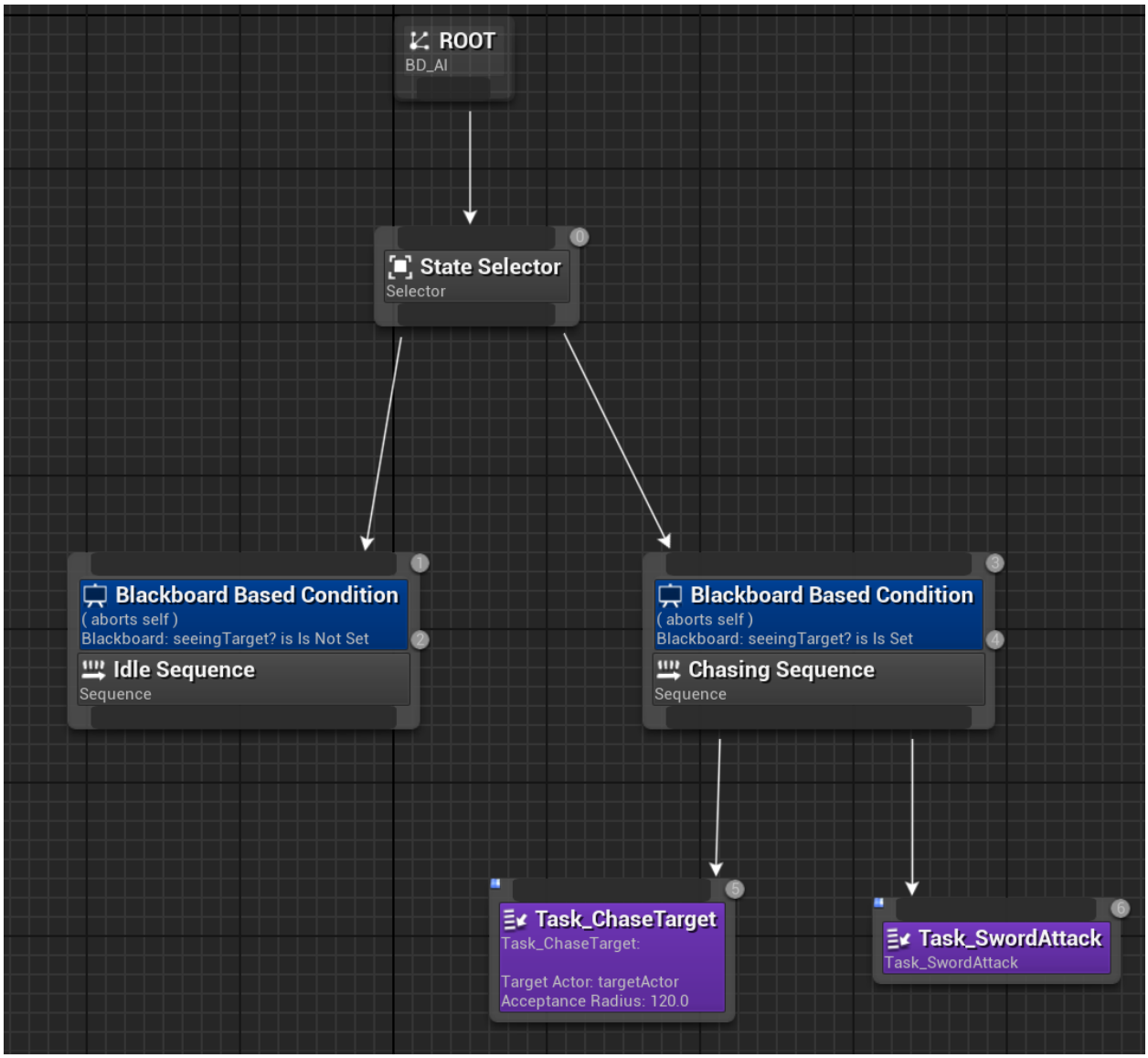

Рис. 14. Дерево поведения вражеского персонажа окружения

Для реализации системы здоровья, комбинаций атак, смерти вражеского персонажа окружения в классе BP_Enemy были созданы соответствующие блоки. Разработанный класс BP_Enemy представлен на рисунке 15.

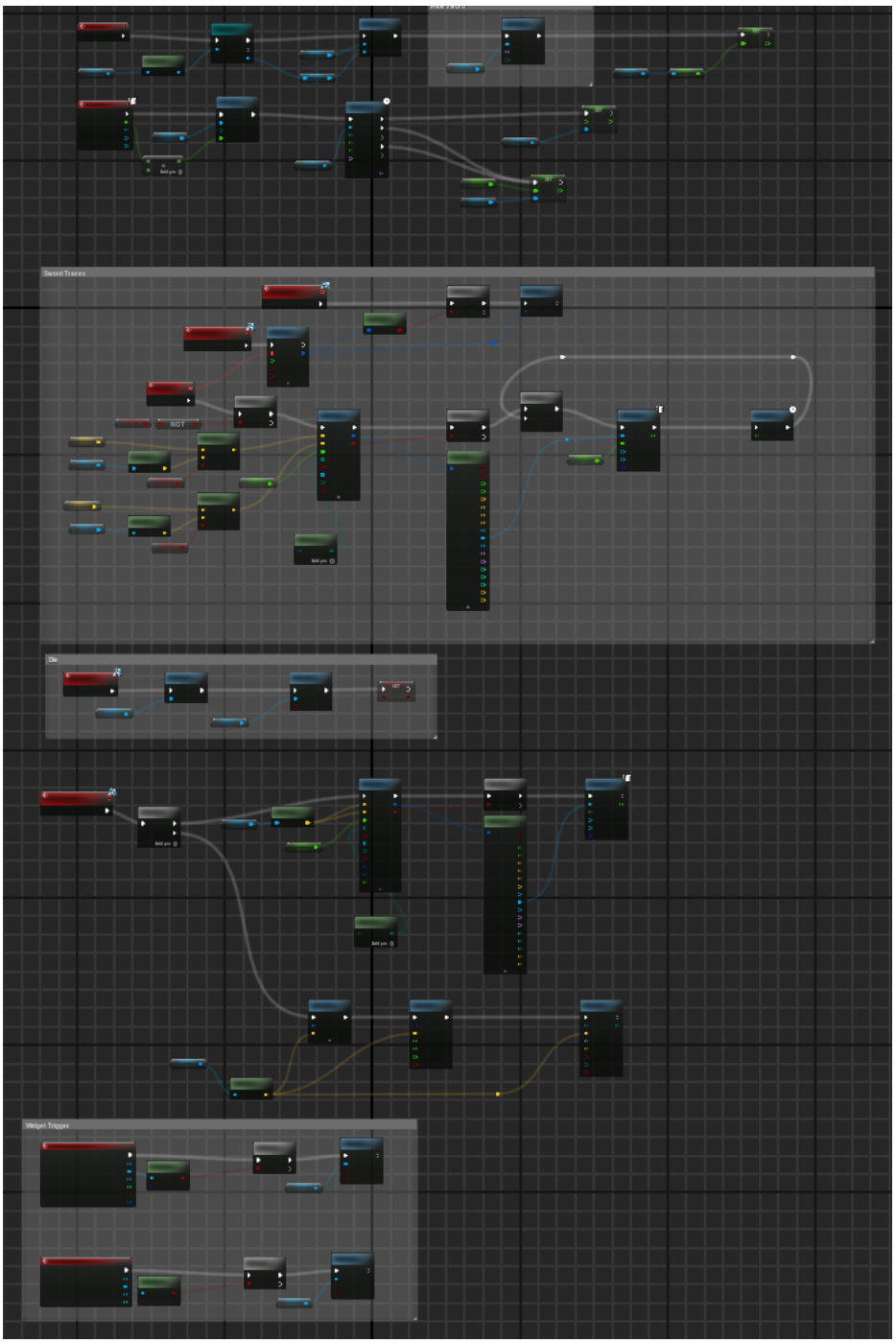

Рис. 15. Класс вражеского персонажа окружения

2.4.6. Главный персонаж окружения

Главный персонаж окружения в сцене создается с целью разнообразия боевых возможностей проекта.

Для его реализации были созданы следующие объекты:

1. BT_Boss – дерево поведения главного персонажа окружения с большим количеством разнообразных действий.

2. BP_Enemy_Boss – основной класс главного персонажа окружения, являющийся дочерним классом BP_Enemy.

Разработанное дерево поведения главного персонажа окружения представлено на рисунке 16.

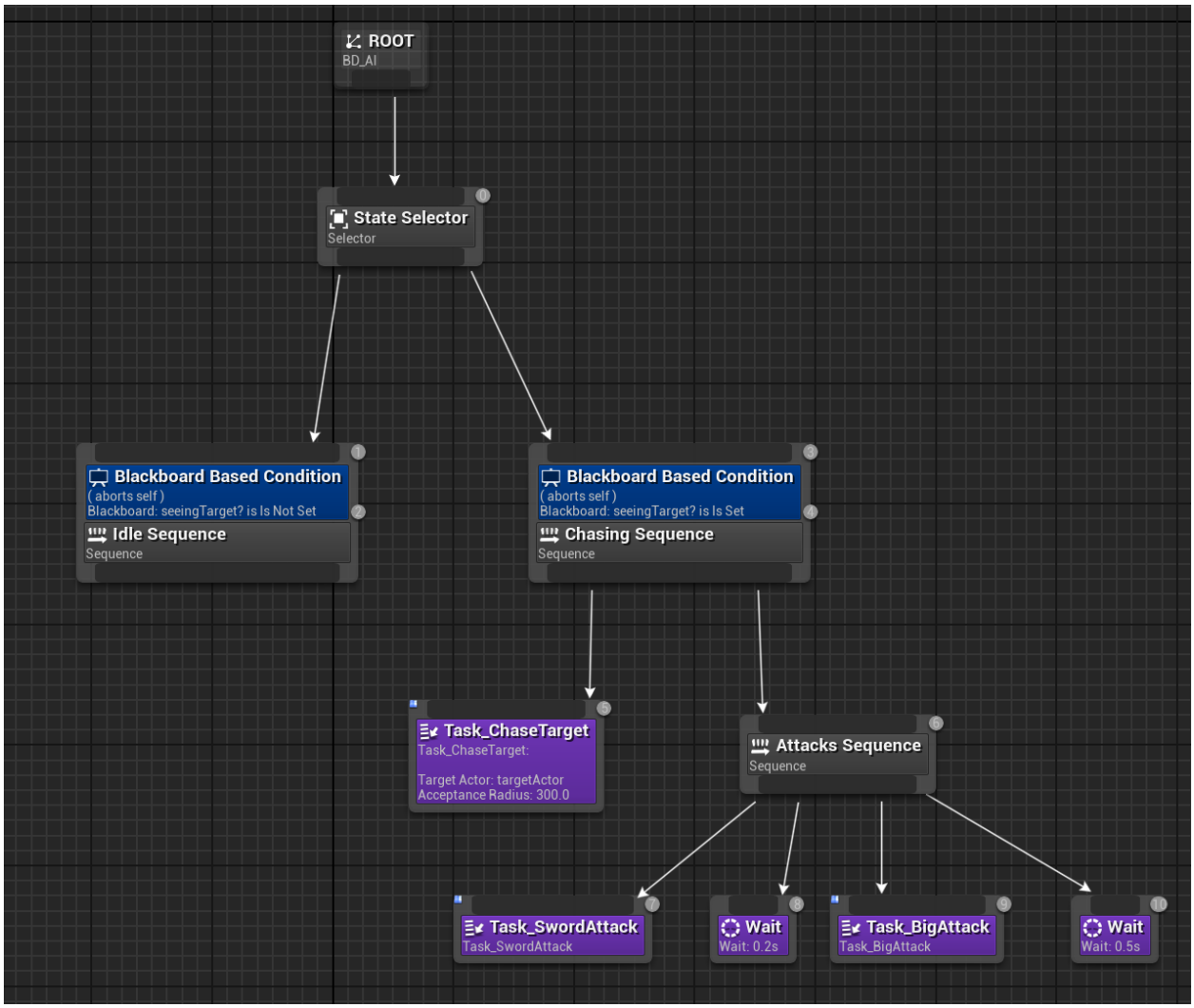

Рис. 16. Дерево поведения главного персонажа окружения

При реализации дополнительной атаки для главного персонажа окружения был создан класс AnimNotify – BP_Notify_PointDamage для оповещения получения урона от данной атаки. Разработанный класс оповещения получения урона представлен на рисунке 17.

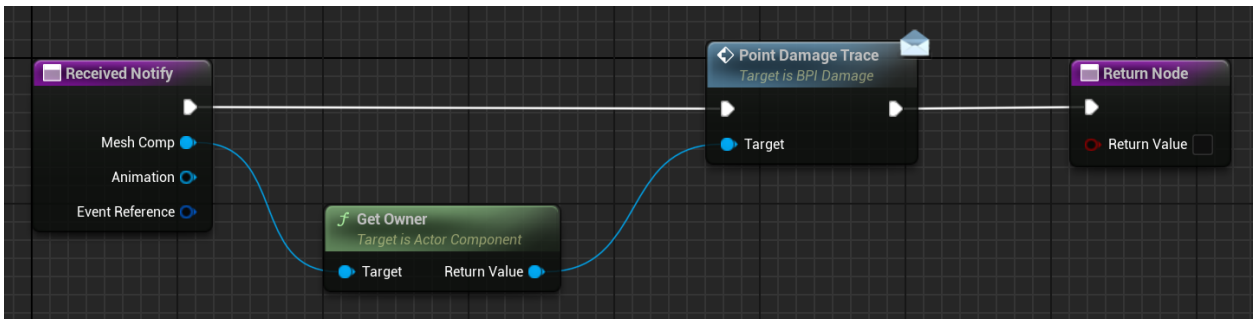

Рис. 17. Класс оповещения получения урона

Разработанный блок для получения урона от атаки главного персонажа окружения представлен на рисунке 18.

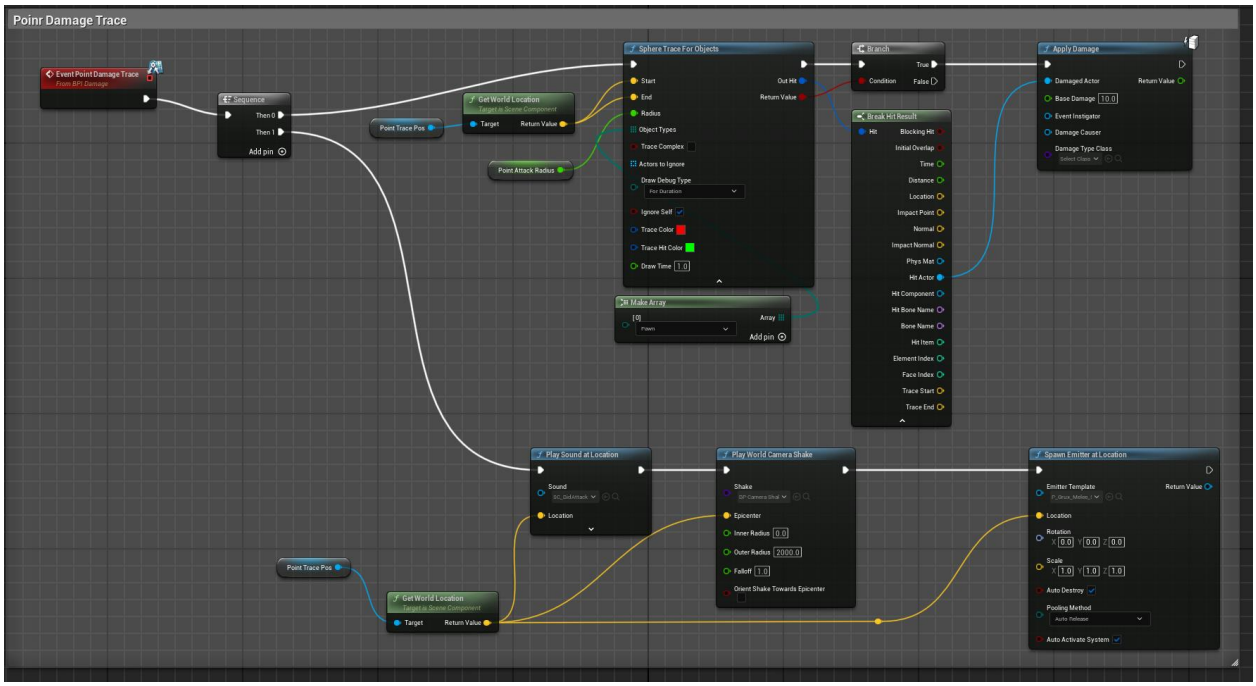

Рис. 18. Блок получения урона от атаки главного персонажа окружения

Для визуализации системы здоровья главного персонажа окружения созданы виджет WB_BossHealth и сфера коллизии Boss Trigger, в области которой происходит визуализация интерфейса здоровья на экране пользователя. Разработанный блок визуализации системы здоровья в сцене представлен на рисунке 19.

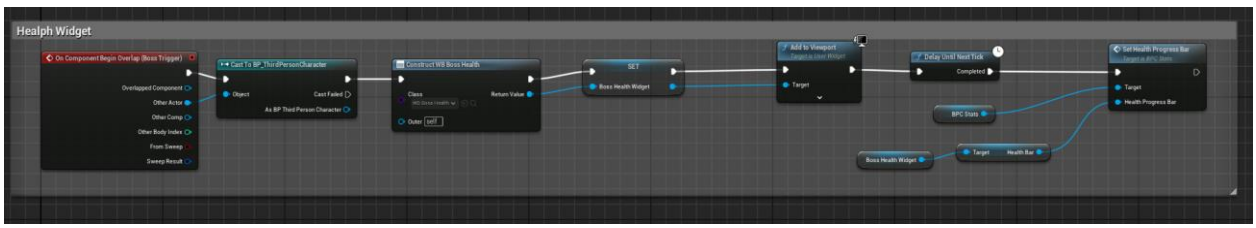

Рис. 19. Визуализация здоровья главного персонажа окружения

ЗАКЛЮЧЕНИЕ

В ходе прохождения практики были получены навыки работы с Unreal Engine 5, получен опыт разработки на языке C++; улучшено представление об организации проектов.

В результате были достигнуты следующие цели:

1. Разработка персонажей окружения;
2. Разработка боевой системы.

При достижении вышеперечисленных целей были решены следующие задачи:

1. Создан ИИ для контроля действий персонажа окружения;
2. Настроены анимации перемещения персонажей;
3. Создана возможность захвата цели;
4. Создана комбинация атак;
5. Реализована система здоровья;
6. Реализована система уклонения;
7. Разработан вражеский персонаж окружения;
8. Разработан главный персонаж окружения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Epic Games Unreal Engine Documentation / Epic Games [Электронный ресурс] // Epic Games Developer: [сайт]. — URL: https://dev.epicgames.com/documentation/ru-ru/unreal-engine/unreal-engine-5-5-documentation (дата обращения: 05.05.2025).

2. Unreal Engine Building natural environments / Unreal Engine [Электронный ресурс] // YouTube: [сайт]. — URL: https://www.youtube.com/watch?v=gbj1qgPOl3E (дата обращения: 05.05.2025).

3. Том Шэннон Unreal Engine 4 для дизайна и визуализации [Текст] / Том Шэннон — 1-е изд. — Москва: Bombora, 2021 — 368 с.

4. Epic Games Unreal Engine / Epic Games [Электронный ресурс] // GitHub: [сайт]. — URL: https://github.com/EpicGames (дата обращения: 7.11.2024).

5. Максименкова Ольга Вениаминовна, Веселко Никита Игоревич Программирование в Unreal Engine 5. Основы визуального языка Blueprint. [Текст] / Максименкова Ольга Вениаминовна, Веселко Никита Игоревич — 1-е изд.. — Москва: Эксмо, 2023 — 320 с.