Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное 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)*

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

НА ТЕМУ

«*Разработка реалистичных природных ландшафтов на Unreal Engine 4*»

Студент РК6-81Б _____ А.В. Фёдоров
(Группа) (подпись, дата) (инициалы и фамилия)

Руководитель КП _____ Ф.А. Витюков
(подпись, дата) (инициалы и фамилия)

Консультант _____ Ф.А. Витюков
(подпись, дата) (инициалы и фамилия)

Нормоконтролер _____ С.В. Грошев
(подпись, дата) (инициалы и фамилия)

Москва, 2023 г.

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

УТВЕРЖДАЮ
Заведующий кафедрой *РК-6*
(индекс)

_____ *А.П. Карпенко*
(инициалы и фамилия)

«___» _____ *202_* г.

З А Д А Н И Е
на выполнение курсового проекта

Студент группы РК6-81Б

Фёдоров Артемий Владиславович
(фамилия, имя, отчество)

Тема курсового проекта

Разработка анимации персонажей и боевой системы на Unreal Engine 4

Источник тематики (кафедра, предприятие, НИР): кафедра

Тема утверждена распоряжением по факультету РК № _____ от «___» _____ *202_* г.

Часть 1. *Аналитический обзор литературы*
В рамках аналитического обзора литературы должен быть изучен полный цикл создания играбельных персонажей. Должны быть изучены средства работы с управлением и анимациями персонажей, предоставляемые движком Unreal Engine 4.

Часть 2. *Практическая часть 1.*
Должна быть разработана система перемещения игрового персонажа с использованием таких способов перемещения как ходьба, бег, ходьба в приседе и скольжение по наклонным поверхностям. Должна быть разработана боевая система, позволяющая игровому персонажу выбирать цели и атаковать их мечом с произвольным направлением замаха и удара.

Часть 3. *Практическая часть 2.*
Должен быть разработан набор анимаций для перемещения персонажа, а также набор анимаций атак, позволяющий реализовать атаки под произвольными углами.

Оформление выпускной квалификационной работы:

Расчетно-пояснительная записка на [8] листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.):

8 графических листов

Дата выдачи задания «01» февраля 2024 г.

Студент А.В. Фёдоров

(Подпись, дата) (И.О.Фамилия)

Руководитель выпускной квалификационной работы Ф.А. Витюков

(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

РЕФЕРАТ

Работа посвящена различным способами и инструментам разработки игровых персонажей в Unreal Engine 4.

В данной работе рассмотрены и проиллюстрированы следующие этапы: Создание системы перемещения персонажа, создание анимаций атак персонажа, импорт в движок Unreal Engine 4, настройка интерполяции позы персонажа между различными наборами анимаций атак, реализация регистрации попадания удара по цели, создание визуального интерфейса для отслеживания направления атак персонажа.

Тип работы: научно-исследовательская работа.

Тема работы: «Разработка анимаций персонажа и боевой системы на Unreal Engine 4».

Объект исследований: 3d-моделирование и анимация, игровые персонажи, шейдеры.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире анимированные персонажи применяются повсеместно в индустрии видеоигр и кинематографа. Как правило — это реалистичные трёхмерные модели людей.

В данной работе рассматривается создание игровых персонажей с использованием трёхмерного движка Unreal Engine 4. Данный инструмент выделяет набор готовых технологий для работы с управляемыми персонажами и трёхмерными анимациями.

Цели работы:

1. Создать систему перемещения персонажа, включающую в себя ходьбу, бег, перемещение в приседе и скольжение по наклонным поверхностям.
2. Создать набор анимаций для перемещения и атак персонажа.
3. Создать боевую систему, позволяющую выбирать цель и атаковать её мечом с произвольным направлением удара.

Для выполнения работы были использованы средства следующих программ:

- Unreal Engine 4 – разработка внутриигровых систем и управление анимациями;
- Blender 3D – создание 3d-анимаций персонажа;

1. Создание системы перемещения персонажа

1.1. Решения, предоставляемые движком Unreal Engine 4

Движок Unreal Engine 4 предоставляет базовый функционал для создания управляемых игровых персонажей. В частности, присутствует набор классов, которые могут быть расширены с помощью языка C++:

AActor – «актёр», базовый класс для любого объекта, который может быть размещен в виртуальном 3д пространстве.

APawn – «пешка», базовый класс для любого «актёра», который может находится под управлением игрока или компьютера.

ACharacter – «пешка», которая обладает «телом», коллайдером (компонентом, отвечающим за столкновение с другими объектами) и базовой логикой перемещения, состоящей из ходьбы и прыжков.

*A*CharacterMovementComponent – класс, отвечающий за логику перемещения персонажа и различные режимы перемещения.

1.2. Программная реализация

Был разработан класс *ACustomCharacter*, наследующий от класса ACharacter. Данный класс служит основой для всех последующих разработанных классов и содержит в себе логику обработки вводов игрока и поля для хранения разнообразных переменных состояния.

Был разработан класс *ACustomCharacterMovementComponent*, наследующий от класса ACharacterMovementComponent. Данный класс отвечает за перемещение объекта класса ACustomCharacter. В частности функция *ACharacterMovementComponent::OnMovementUpdated()* были перегружена для возможности управление максимальной скоростью персонажа в зависимости от режима перемещения и была разработана функция *ACustomCharacterMovementComponent::PhysSlide()* для расчёта «физического» перемещения персонажа при скольжении по наклонной поверхности.

Листинг 1. Функция *ACustomCharacterMovementComponent::PhysSlide()*, отвечающая за логику перемещения персонажа в состоянии скольжения.

```
void UCustomCharacterMovementComponent::PhysSlide(float deltaTime, int32
Iterations)
{
	if (deltaTime < MIN_TICK_TIME) { return; }

	FHitResult SurfaceHit;
	if
(!GetSlideSurface(SurfaceHit)||Velocity.SizeSquared()<pow(Slide_MinSpeed, 2))
	{
		ExitSlide();
		StartNewPhysics(deltaTime, Iterations);
		return;
	}

	Velocity += Slide_GravityForce * FVector::DownVector * deltaTime;

	if (FMath::Abs(FVector::DotProduct(Acceleration.GetSafeNormal(),
UpdatedComponent->GetRightVector())) > 0.5)
	{
		Acceleration = Acceleration.ProjectOnTo(UpdatedComponent-
>GetRightVector());
	}
	else
	{
		Acceleration = FVector::ZeroVector;
	}

	if (!HasAnimRootMotion() && !CurrentRootMotion.HasOverrideVelocity())
	{
		CalcVelocity(deltaTime, Slide_Friction, true,
GetMaxBrakingDeceleration());
	}
	ApplyRootMotionToVelocity(deltaTime);

	Iterations++;
	bJustTeleported = false;

	FVector OldLocation = UpdatedComponent->GetComponentLocation();
	FQuat OldRotation = UpdatedComponent-
>GetComponentRotation().Quaternion();
	FHitResult Hit(1.f);
	FVector Adjusted = Velocity * deltaTime;
	FVector VelPlaneDir = FVector::VectorPlaneProject(Velocity,
SurfaceHit.Normal).GetSafeNormal();
	FQuat NewRotation = FRotationMatrix::MakeFromXZ(VelPlaneDir,
SurfaceHit.Normal).ToQuat();

	SafeMoveUpdatedComponent(Adjusted, NewRotation, true, Hit);

	if (Hit.Time < 1.f)
	{
		HandleImpact(Hit, deltaTime, Adjusted);
		SlideAlongSurface(Adjusted, (1.f - Hit.Time), Hit.Normal, Hit,
true);
	}

	FHitResult NewSurfaceHit;
	if (!GetSlideSurface(NewSurfaceHit) || Velocity.SizeSquared() <
pow(Slide_MinSpeed, 2))
	{
```

```
            ExitSlide();
      }

      if (!bJustTeleported && !HasAnimRootMotion() &&
!CurrentRootMotion.HasOverrideVelocity())
      {
            Velocity = (UpdatedComponent->GetComponentLocation() -
OldLocation) / deltaTime;
      }
}
```

Листинг 2. Функция *ACustomCharacterMovementComponent:: OnMovementUpdated()*, отвечающая за изменение максимальной скорости персонажа в зависимости от режима перемещения.

```
void UCustomCharacterMovementComponent::OnMovementUpdated(float DeltaSeconds,
const FVector& OldLocation,
                                                            const FVector&
OldVelocity)
{
      Super::OnMovementUpdated(DeltaSeconds, OldLocation, OldVelocity);

      if (MovementMode == MOVE_Walking && !IsCrouching())
      {
            if(CustomCharacterOwner->IsDoingAnAttack())
            {
                  MaxWalkSpeed=Walk_MaxAttackWalkSpeed;
            }
            else if (Safe_bWantsToSprint) // &&
!GetCurrentAcceleration().IsZero()
            {
                  MaxWalkRunSpeed = FMath::FInterpTo(MaxWalkRunSpeed,
Sprint_MaxWalkSpeed, DeltaSeconds, Walk_Sprint_InterpSpeedUp);
                  MaxWalkSpeed = MaxWalkRunSpeed;
            }
            else
            {
                  MaxWalkRunSpeed= FMath::FInterpTo(MaxWalkRunSpeed,
Walk_MaxWalkSpeed, DeltaSeconds, Walk_Sprint_InterpSpeedDown);
                  MaxWalkSpeed = MaxWalkRunSpeed;
            }
      }
      if (IsCrouching() && !IsSliding())
      {
            MaxWalkRunSpeed = FMath::FInterpTo(MaxWalkRunSpeed,
Walk_MaxWalkSpeed, DeltaSeconds, Walk_Sprint_InterpSpeedDown);
      }else
      {

      }
      bPrevWantsToCrouch = bWantsToCrouch;
}
```

2. Создание анимированного персонажа

При анимации персонажей речь всегда идёт о скелетной анимации (Skeletal Animations). Для модели персонажа создаётся скелет, и анимация состоит из вращения костей друг относительно друга.

2.1.Решения, предоставляемые движком Unreal Engine 4

В базовых ассетах Unreal Engine 4 присутствует минималистичная модель человека и небольшой набор анимаций, связанных с ней. Все созданные далее анимации будут применяться к данной модели.

Рисунок 1. Базовая модель человека, поставляемая с Unreal Engine 4.

Технология *Animation Blueprints* позволяет управлять логикой анимаций конкретной модели с помощью визуального программирования. Возможно выполнять плавные переходы между анимациями, напрямую управлять костями скелета или настраивать логику, которая в конечном итоге будет определять окончательную позу модели для каждого кадра.

Технология *State Machines*, включенная в *Animation Blueprints*, предоставляет способ разбить анимацию персонажа на набор различных состояний. Эти состояния затем регулируются *Transition Rules* (Правилами

перехода), которые контролируют, как переходить из одного состояния в другое. *State Machines* значительно упрощает анимацию персонажей, которые могут выполнять большое количество несовместных действий, таких как ходьба, бег и прыжки.

Blendspaces – функции, предоставляемые движком Unreal Engine 4, позволяющие плавно интерполировать позу персонажа между рядами заданных анимаций.

2.2.Настройка анимаций перемещения

С помощью технологии *Blendspaces* был создан континуум различных анимаций ходьбы, позволяющий персонажу плавно переходить из стационарного положения в ходьбу, затем в легкую трусцу и наконец в полноценный бег.

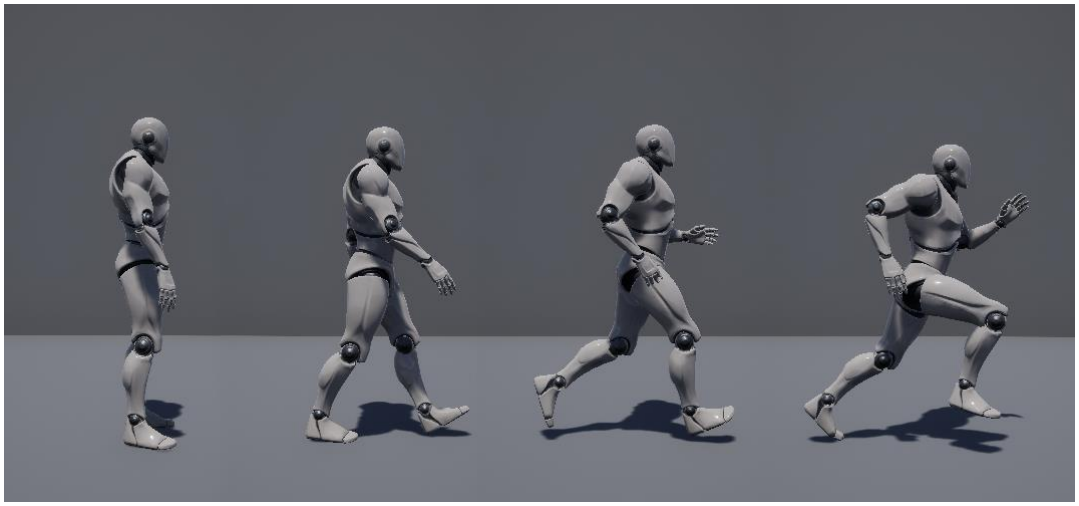

Рисунок 2. Выбор анимации в зависимости от скорости персонажа.

С помощью *State Machines* был создан набор состояний, позволяющих управлять анимацией скольжения персонажа по наклонной поверхности. Это процесс поделен на пять этапов: бег, вхождение в скольжение, скольжение, выход из скольжения и снова бег. Персонаж переходит между этими состояниями в зависимости от вводов игрока и времени, проведенном в скольжении.

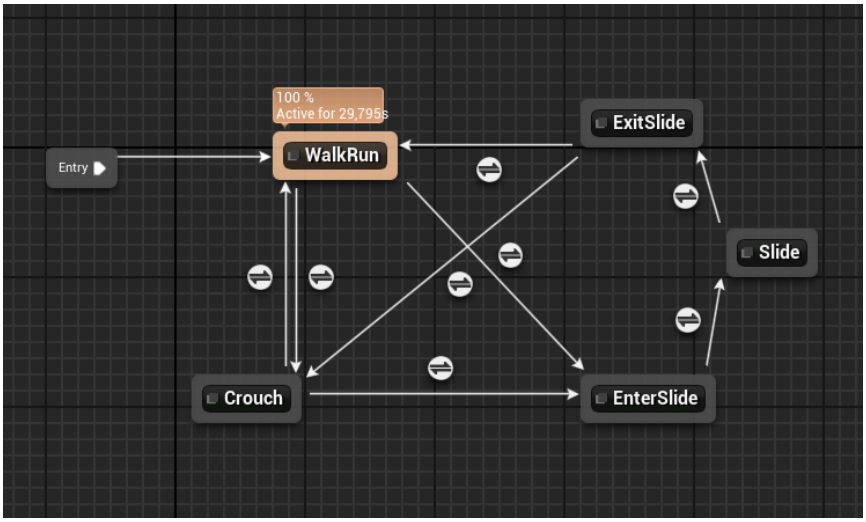

Рисунок 3. Набор состояний, определяющих анимацию персонажа во время скольжения по наклонной поверхности.

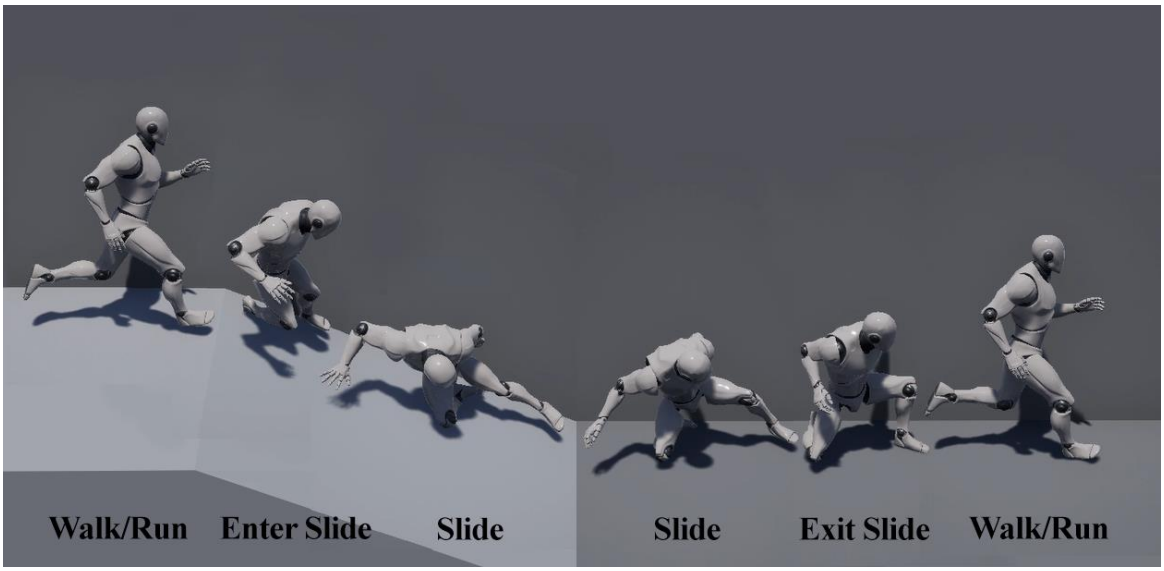

Рисунок 4. Демонстрация работы машины состояний.

2.3.Создание анимаций атак

Для возможности выбирать игроком произвольный угол атаки было принято решение сделать набор анимаций атак, между которыми будет проводится интерполяция.

С помощью программы Blender 3D были разработаны анимации горизонтального удара слева направо, вертикального удара сверху вниз и горизонтального удара справа налево. Такой набор позволяет получить анимацию любого удара, который проводится сверху вниз с помощью интерполяции.

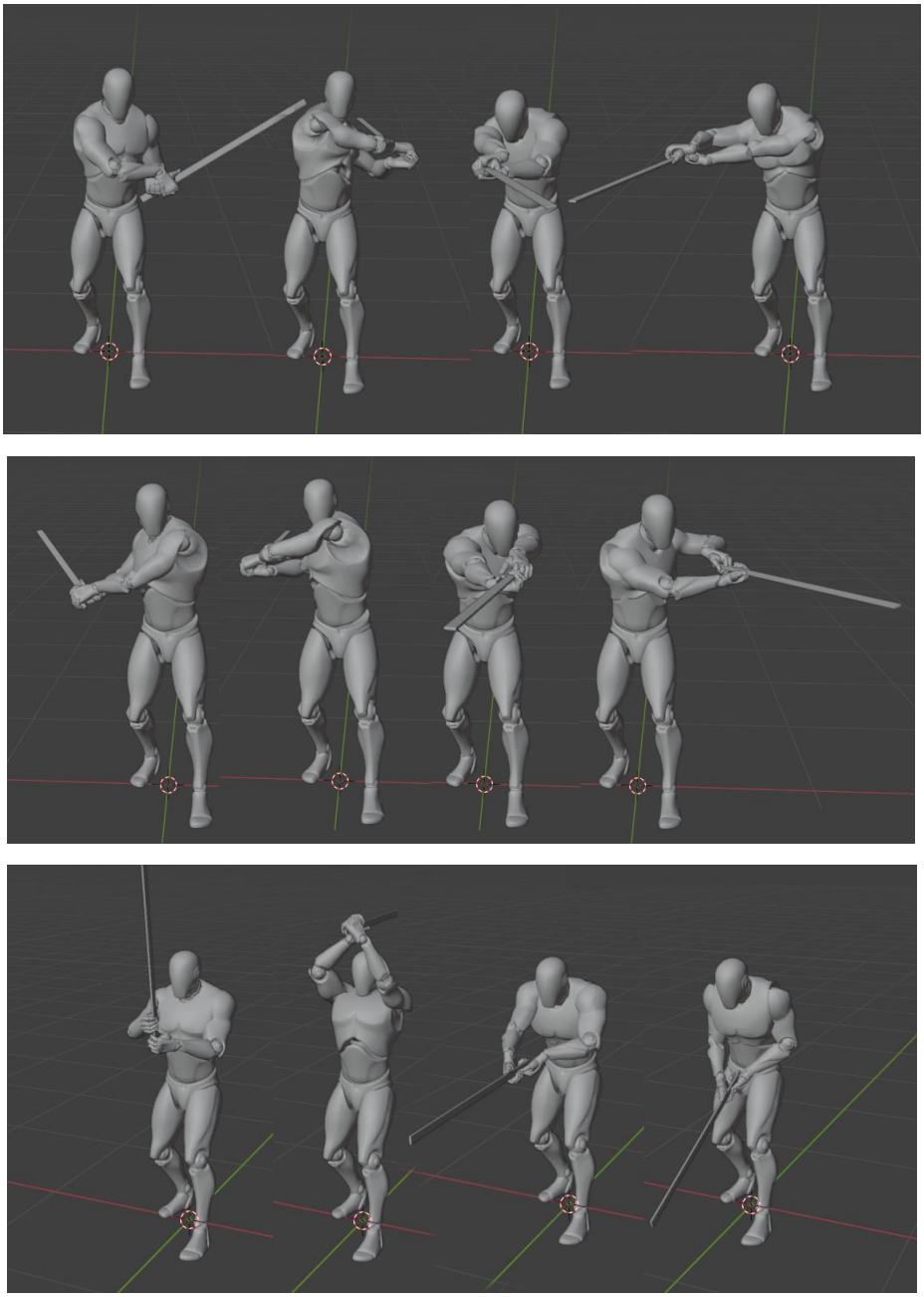

Рисунок 5. Наборы анимаций ударов, созданных в программе Blender 3d.

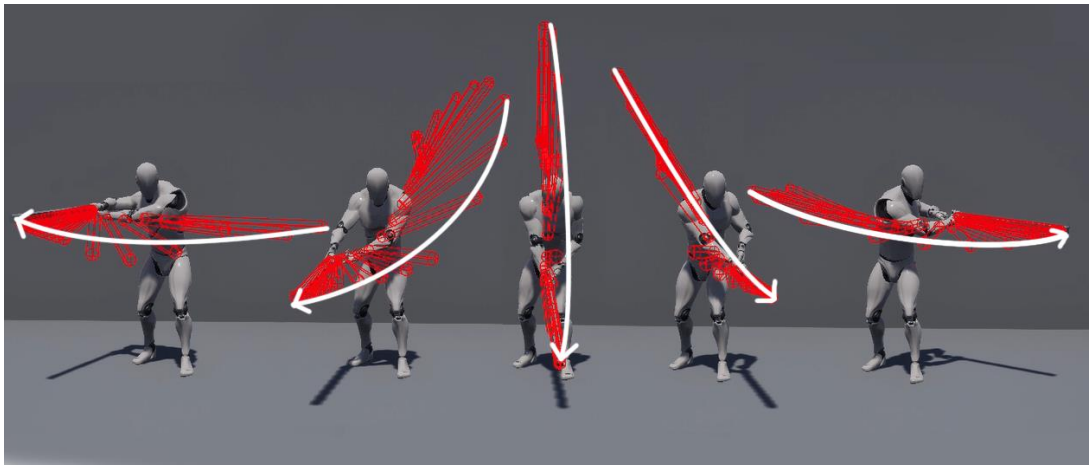

Рисунок 6. Атаки различных направлений, полученные с помощью интерполяций поз.

3. Разработка боевой системы

Для наглядного выбора цели была реализовано перемещение камеры «за плечо» персонажа по направлению к цели. Для этого был разработан класс *ACustomPlayerManager*, наследующий от класса *APlayerManager*, отвечающего за управление камерой. В разработанном классе была перегружена функция APlayerManager::UpdateViewTarget()

Листинг 3. Функция ACustomPlayerManager::UpdateViewTarget(), отвечающая за положение камеры за плечом персонажа.

```
void ACustomPlayerCameraManager::UpdateViewTarget(FTViewTarget& OutVT, float
DeltaTime)
{
	Super::UpdateViewTarget(OutVT, DeltaTime);
	if(ACustomCharacter* CustomCharacter =
Cast<ACustomCharacter>(GetOwningPlayerController()->GetPawn()))
	{
		UCustomCharacterMovementComponent* CMC = CustomCharacter-
>GetCustomCharacterMovementComponent();

		FVector TargetCrouchOffset = FVector(0,
			0,
			CMC->CrouchedHalfHeight - CustomCharacter->GetClass()-
>GetDefaultObject<ACharacter>()->GetCapsuleComponent()-
>GetScaledCapsuleHalfHeight()
			);
			FVector Offset =
FMath::InterpEaseInOut(FVector::ZeroVector,TargetCrouchOffset,FMath::Clamp(Cro
uchBlendTime/CrouchBlendDuration,0.f,1.f),TransitionEaseInOutExponent);
		if(CustomCharacter->IsLockedOn())
		{
			AActor* Target=CustomCharacter->GetTargetedActor();
			checkf(Target!=nullptr,TEXT("CustomPlayerCameraManager:
bIsLockedOn is true, but targeted actor is missing"));
			FVector toTarget=Target->GetActorLocation()-CustomCharacter-
>GetActorLocation();
			FVector absOffset=lockOnOffset;
			if(CustomCharacter->IsShoulderCameraFlipped())absOffset.Y=-
absOffset.Y;
			FVector
relativeShoulderCamOffset=toTarget.Rotation().RotateVector(absOffset);
			FVector lookAtPosition=CustomCharacter-
>GetActorLocation()+relativeShoulderCamOffset;
			FRotator
lookAtRotation=UKismetMathLibrary::FindLookAtRotation(GetCameraLocation(),((Ta
rget->GetActorLocation()+CustomCharacter->GetActorLocation())/2));

	OutVT.POV.Location=FMath::VInterpTo(GetCameraLocation(),lookAtPosition,D
eltaTime,LockOnFollowSpeed);

	OutVT.POV.Rotation=FMath::RInterpTo(GetCameraRotation(),lookAtRotation,D
eltaTime,LockOnFollowSpeed);
			GetOwningPlayerController()-
>SetControlRotation(OutVT.POV.Rotation);
```

```
            }

            if(CMC->IsCrouching())
            {
                  CrouchBlendTime =
FMath::Clamp(CrouchBlendTime+DeltaTime,0.f,CrouchBlendDuration);
                  Offset -= TargetCrouchOffset; //?? TEST THIS
            }
            else
            {
                  CrouchBlendTime = FMath::Clamp(CrouchBlendTime-
DeltaTime,0.f,CrouchBlendDuration);
            }

            if(CMC->IsMovingOnGround())
            {
                  OutVT.POV.Location+=Offset;
            }

      }
}
```

Рисунок 7. Расположение камеры за плечом персонажа при выборе цели.

С помощью технологии Notify States, позволяющей выполнять логику в определенный промежуток анимации реализуется удара. Каждый кадр атаки происходит трассировка цилиндром по всей длине меча. При попадании по другому персонажу срабатывает логика, отвечающая за получение урона.

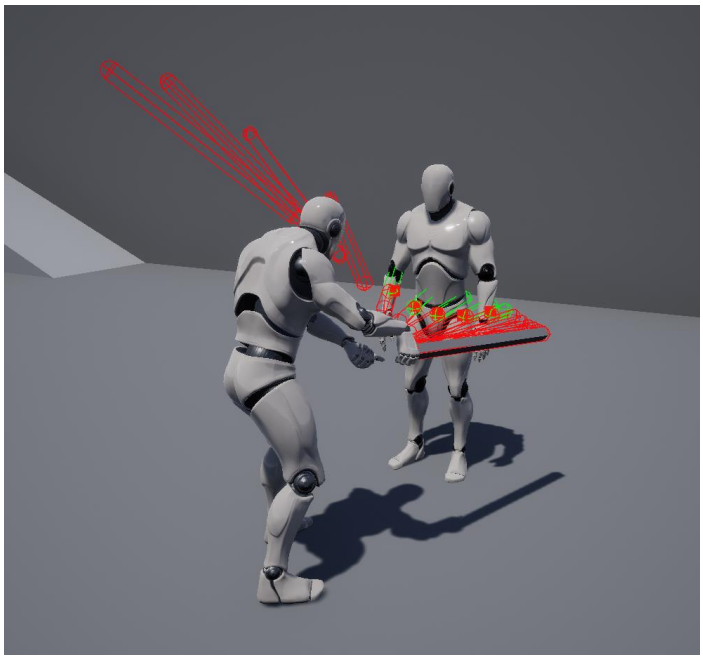

Рисунок 8. Трассировка атаки. Жёлтые фигуры показывают успешное попадание по выбранному персонажу.

ЗАКЛЮЧЕНИЕ

В данной работе были рассмотрены методы создания игровых персонажей с использованием трёхмерного движка Unreal Engine 4.

1. Создана система перемещения персонажа, включающая в себя ходьбу, бег, перемещение в приседе и скольжение по наклонным поверхностям.
2. Созданы и настроены анимации для перемещения и атак персонажа.
3. Создана боевая система, позволяющая выбирать цель и атаковать её мечом с произвольным направлением удара.

СПИСОК ЛИТЕРАТУРЫ

1. Божко А.Н., Жук Д.М., Маничев В.Б. Компьютерная графика. [Электронный ресурс] // Учебное пособие для вузов. − М.: Изд-во МГТУ им. Н. Э. Баумана, 2007. - 389 с., - ISBN 978-5-7038-3015-4, Режим доступа: http://ebooks.bmstu.ru/catalog/55/book1141.html. Дата обращения: 10.02.2024.
2. Unreal Engine 4 Documentation // Unreal Engine Documentation URL: https://docs.unrealengine.com/. Дата обращения: 07.04.2024.
3. Animating Characters and Objects // Unreal Engine Documentation URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/animating-characters-and-objects-in-unreal-engine?application_version=5.2. Дата обращения: 07.04.2024.
4. Animation & Rigging – Blender Manual // Blender Manual URL: https://docs.blender.org/manual/en/latest/animation/index.html. Дата обращения: 18.03.2024.
5. Modeling – Blender Manual // Blender Manual URL: https://docs.blender.org/manual/en/latest/modeling/index.html. Дата обращения: 18.02.2022.
6. Programming Quick Start // Unreal Engine Documentation URL: https://docs.unrealengine.com/5.0/en-US/unreal-engine-cpp-quick-start/. Дата обращения: 29.12.2024.
7. Real-Time Character Animation Techniques // Image Synthesis Group Trinity College Dublin. URL: https://publications.scss.tcd.ie/tech-reports/reports.00/TCD-CS-2000-06.pdf Дата обращения: 05.03.2024.
8. An Indie Approach To Procedural Animation // Wolfire Games. URL: https://gdcvault.com/play/1020049/Animation-Bootcamp-An-Indie-Approach Дата обращения: 05.03.2024
9. Physical Animation in ‘Star Wars Jedi: Fallen Order’ // Respawn Entertainment. URL: https://gdcvault.com/play/1026848/Physical-Animation-in-Star-Wars Дата обращения: 05.03.2024

10. An Inside Look At Fighting Steelrising Animations Made With 3ds Max // 80lv. URL: https://80.lv/articles/an-inside-look-at-fighting-steelrising-animations-made-with-3ds-max/ Дата обращения: 05.03.2024

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

АКТ
проверки выпускной квалификационной работы

Студент группы *РК6-81Б*

Фёдоров Артемий Владиславович

(Фамилия, имя, отчество)

Тема выпускной квалификационной работы: Разработка реалистичных природных ландшафтов на Unreal Engine 4

Выпускная квалификационная работа проверена, размещена в ЭБС «Банк ВКР» в полном объеме и соответствует / не соответствует требованиям, изложенным в Положении о порядке

ненужное зачеркнуть

подготовки и защиты ВКР.

Объем заимствования составляет _____% текста, что с учетом корректного заимствования соответствует / не соответствует требованиям к ВКР

_____

ненужное зачеркнуть *бакалавра, специалиста, магистра*

Нормоконтролёр _____ С.В. Грошев

(подпись) (ФИО)

Согласен:

Студент _____ А.В. Фёдоров

(подпись) (ФИО)

Дата: