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 метода повышения производительности рендеринга групп 3d-объектов на Unreal Engine 4»

Студент РК6-81Б _____ **Афиногенов М.А.**
(Группа) (Подпись, дата) (Фамилия И.О.)

Руководитель ВКР _____ **Витюков Ф.А.**
(Подпись, дата) (Фамилия И.О.)

Нормоконтролёр _____ **Грошев С.В.**
(Подпись, дата) (Фамилия И.О.)

2024 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 РК6
(Индекс)
_____ Карпенко А.П.
(Фамилия И.О.)
«15» февраля 2022 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра

Студент группы РК6-81Б

Афиногенов Михаил Алексеевич
(фамилия, имя, отчество)

Тема квалификационной работы: «Разработка метода повышения производительности рендеринга групп 3d-объектов на Unreal Engine 4»

Источник тематики (НИР кафедры, заказ организаций и т.п.): кафедра.

Тема квалификационной работы утверждена распоряжением по факультету РК №___ от «__»
_____ 2021 г.

Техническое задание

Часть 1. *Аналитическая часть*

Изучить полный цикл создания статических 3d-моделей. Изучить средства разработки, предоставляемые трехмерным движком Unreal Engine 4. Провести продуктовое исследование существующих компьютерных коллекционных карточных игр. Провести анализ реализации объекта «рука» в существующих продуктах.

Часть 2. *Практическая часть 1. 3d-моделирование*

Создать 3d-модели карт для компьютерной карточной игры, пройдя полный цикл создания статических моделей, провести их оптимизацию. Выполнить экспорт ассетов и их импорт в движок Unreal Engine 4.

Часть 3. *Практическая часть 2. Разработка внутриигровых систем. Оптимизация рендеринга*

Разработать систему выкладки карт на поле боя. Разработать систему «руки» игрока. Оптимизировать отрисовку большого количества однообразных объектов. Разработать систему пулов отрисовки (Render Pools).

Оформление выпускной квалификационной работы

Расчетно-пояснительная записка на 76 листах формата А4.

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

Работа содержит 7 графических листов формата А4.

Дата выдачи задания: «10» февраля 2022 г.

В соответствии с учебным планом выпускную квалификационную работу выполнить в полном объеме в срок до «21» июня 2024 г.

Руководитель квалификационной работы _____ **Витюков Ф.А.**
(Подпись, дата) (Фамилия И.О.)

Студент _____ **Афиногенов М.А.**
(Подпись, дата) (Фамилия И.О.)

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

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 РК

КАФЕДРА РК6

ГРУППА РК6-81Б

УТВЕРЖДАЮ

Заведующий кафедрой РК6
(Индекс)

_____ Карпенко А.П.
(Фамилия И.О.)

«15» февраля 2022 г.

КАЛЕНДАРНЫЙ ПЛАН
выполнения выпускной квалификационной работы

Студент группы РК6-81Б

Афиногенов Михаил Алексеевич
(фамилия, имя, отчество)

Тема квалификационной работы: «Разработка метода повышения производительности рендеринга групп 3d-объектов на Unreal Engine 4»

№ п/п	Наименование этапов выпускной квалификационной работы	Сроки выполнения этапов		Отметка о выполнении	
		план	факт	Должность	ФИО, подпись
1.	Задание на выполнение работы. Формулирование проблемы, цели и задач работы	10.02.2022	_____	Руководитель ВКР	Витюков Ф.А.
2.	1 часть. Аналитическая часть	18.02.2022	_____	Руководитель ВКР	Витюков Ф.А.
3.	Утверждение окончательных формулировок решаемой проблемы, цели работы и перечня задач	28.02.2022	_____	Заведующий кафедрой	Карпенко А.П.
4.	2 часть. Практическая часть 1	21.04.2022	_____	Руководитель ВКР	Витюков Ф.А.
5.	3 часть. Практическая часть 2	23.05.2022	_____	Руководитель ВКР	Витюков Ф.А.
6.	1-я редакция работы	28.05.2022	_____	Руководитель ВКР	Витюков Ф.А.
7.	Подготовка доклада и презентации	04.06.2022	_____	Студент	Котельникова Е.Ю.
8.	Заключение руководителя	10.06.2022	_____	Руководитель ВКР	Витюков Ф.А.
9.	Допуск работы к защите на ГЭК (нормоконтроль)	17.06.2022	_____	Нормоконтролер	Грошев С.В.
10.	Внешняя рецензия	17.06.2022	_____		
11.	Защита работы на ГЭК	21.06.2024	_____		

Студент _____ Афиногенов М.А. Руководитель ВКР _____ Витюков Ф.А.
(подпись, дата) (Фамилия И.О.) (подпись, дата) (Фамилия И.О.)

АННОТАЦИЯ

В данной работе рассмотрены подходы для создания некоторых элементов пользовательского интерфейса для сбора и редактирования информации о фондовых рынках. Описана разработка трехмерных интерактивных графических элементов. Рассмотрена реализация редактирования графических элементов. Также описаны этапы оптимизации 3D-моделей интерактивных элементов для дальнейшего использования в движке. Проведен анализ эффективности использования компонентной реализации, выполнено сравнение производительности реализующих его компонентов в движке Unreal Engine 4.

В расчётно-пояснительной записке 32 страницы, 4 рисунка, 4 графических листа.

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В последние годы разработка интерактивных элементов пользовательского интерфейса (UI) стала неотъемлемой частью создания современных видеоигр и приложений виртуальной реальности. Популярность трехмерного движка Unreal Engine 4 значительно возросла, поскольку этот движок предоставляет широкий спектр инструментов и функций для создания высококачественных и интерактивных UI.

В данной научно-исследовательской работе будут рассмотрены методы разработки интерактивных элементов пользовательского интерфейса в Unreal Engine 4. В начале будет представлен обзор основных компонентов UE4, которые необходимы для создания UI, а затем будут приведены примеры различных методов разработки, включая использование скриптов, редактирование XML и создание пользовательских компонентов. Перед тем как начать разработку UI в UE4, необходимо ознакомиться с основными компонентами движка.

Один из ключевых компонентов — это Viewport, который представляет собой окно, отображающее игровой мир. В этом окне пользователь может взаимодействовать с объектами и элементами UI.

Другой важный компонент — это HUD (Head-Up Display), который отображает информацию о игре. HUD обычно располагается в верхней части экрана и содержит различные элементы, такие как кнопки и шкалы. Для создания интерактивных элементов UI в UE4 можно использовать несколько методов. Один из них — это использование скриптов. Скрипты позволяют программистам создавать сложные функции и поведения для элементов UI. Например, можно написать скрипт, который реагирует на нажатие кнопки и изменяет цвет фона или анимацию объекта.

Еще один метод разработки UI — это редактирование XML. В UE4 все элементы UI описываются в формате XML. Это позволяет разработчикам создавать сложные макеты и настраивать поведение элементов. Например, можно создать XML-файл, который определяет расположение и размер кнопок на экране.

Кроме того, в UE4 можно создавать пользовательские компоненты. Это позволяет разработчикам создавать уникальные элементы UI, которые соответствуют специфическим требованиям проекта. Например, можно создать пользовательский компонент, который отображает текстовые сообщения или анимации.

Выполнение данного курсового проекта подразумевает под собой разработку, проектировку и реализацию в виде кода части функционала сайта tradingview.com. Далее показано возможное расположение и отображение необходимых элементов на сайте tradingview.com [5] (рис. 1)

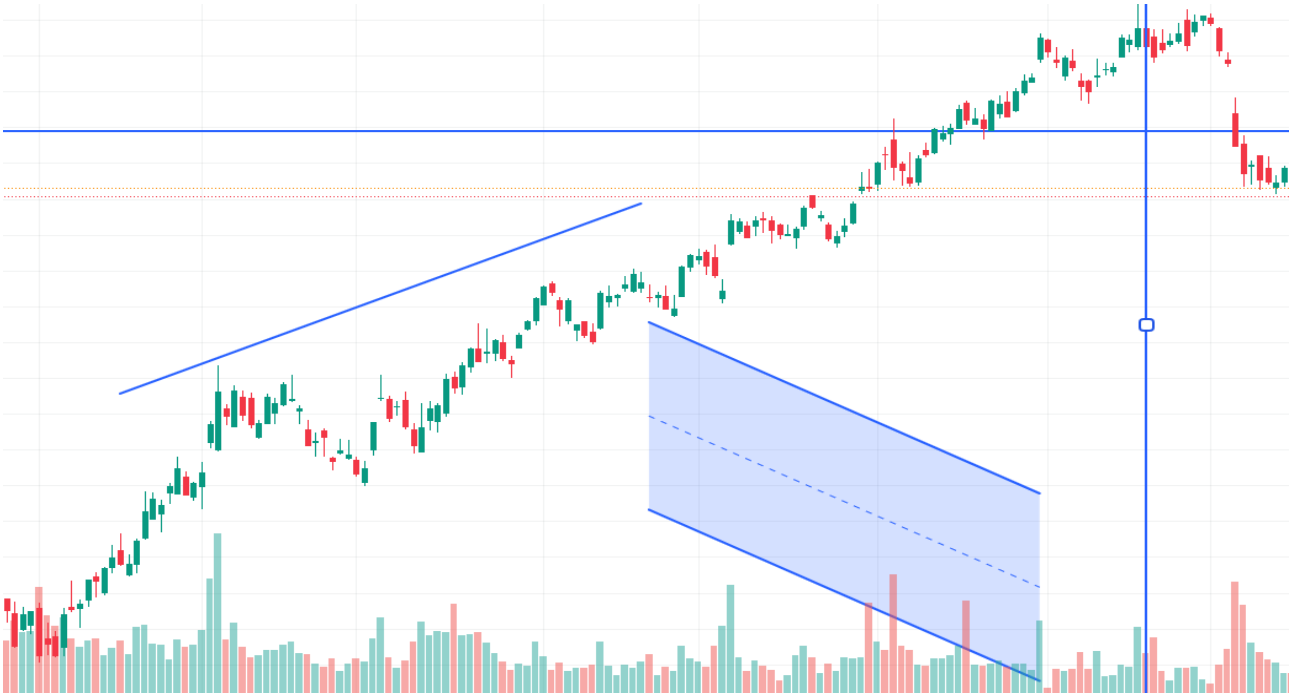

Рисунок 1. Расположение необходимых объектов на сайте tradingview.com [5]

В ходе данного курсового проекта те же самые функциональные элементы были реализованы на базе игрового движка Unreal Engine 4 (рис. 2)

Рисунок 2. Расположение необходимых объектов на Unreal Engine 4

Следует отметить, что разработка интерактивных элементов пользовательского интерфейса в Unreal Engine 4 является достаточно простой благодаря широкому спектру инструментов и функций, предоставляемых движком. Использование скриптов, редактирование XML и создание пользовательских компонентов — это лишь некоторые из методов разработки UI в UE4. Каждый метод имеет свои преимущества и недостатки, поэтому выбор метода зависит от конкретных требований проекта.

1. Обзор существующих решений

В разработке интерактивных элементов пользовательского интерфейса (UI) для трехмерных видеоигр и приложений виртуальной реальности с помощью движка Unreal Engine 4 используются различные методы и подходы. Основным компонентом, используемым для создания интерактивных объектов, является Static Mesh Component, который позволяет прикреплять к объектам классов AActor (далее акторам) или объектам в игровом мире детальную информацию о геометрии и материалах объекта.

Подходы к созданию интерактивности с использованием Static Mesh Component варьируются. Один из них включает использование уже готовых моделей и анимаций. В Unreal Engine 4 существует обширный набор таких моделей и анимаций, которые можно адаптировать для создания интерактивных элементов UI. Например, можно использовать модель двери с анимацией открытия и закрытия, чтобы создать кнопку-переключатель.

Второй подход включает создание собственных моделей и анимаций. Unreal Engine 4 предлагает мощные инструменты для создания и редактирования геометрии объектов и анимаций. Это позволяет разработчикам создавать уникальные интерактивные элементы UI, полностью соответствующие требованиям их проектов.

Третий подход включает программирование интерактивности с использованием Blueprint — визуального языка программирования в Unreal Engine 4. С помощью Blueprint разработчики могут легко добавлять логику взаимодействия с интерактивными элементами UI. Например, можно программировать обработку нажатия на кнопку или изменение значения ползунка.

Кроме того, Unreal Engine 4 позволяет использовать физическую симуляцию для создания более реалистичной интерактивности. Например, при нажатии на кнопку она может отталкиваться от пальца пользователя или перемещаться по заданной траектории.

В целом, методы разработки трехмерных интерактивных элементов и моделей на движке движке Unreal Engine 4 обширны и позволяют достигать

высокого уровня реализма и функциональности. Они предоставляют возможность создания уникальных и привлекательных игровых интерфейсов, которые могут улучшить впечатление пользователей от игры.

Эти методы включают в себя не только технические аспекты создания интерактивных элементов, но и учет пользовательского опыта и интерфейсного дизайна. Понимание того, как пользователи взаимодействуют с трехмерным интерфейсом, и интеграция этого понимания в процесс разработки являются ключевыми для создания эффективных и интуитивно понятных пользовательских интерфейсов. Это требует тщательного анализа пользовательского поведения и предпочтений, а также тестирования и итеративного улучшения интерфейса на основе обратной связи от пользователей.

Разработка интерактивных элементов UI в трехмерном пространстве представляет собой сложную, но в то же время захватывающую задачу, которая открывает новые возможности для разработчиков и дизайнеров в области создания виртуальных миров и игровых интерфейсов. Благодаря инновационным подходам и технологиям, таким как Unreal Engine 4, сфера разработки видеоигр и виртуальной реальности продолжает развиваться, предоставляя все более мощные и эффективные инструменты для создания впечатляющих и вовлекающих пользовательских интерфейсов.

2. Оптимизация отрисовки трехмерных объектов

Одним из важных аспектов разработки UI в UE4 является оптимизация отображения трехмерных элементов. Для этого используются различные техники, включая Static Mesh Component, Instance Static Mesh и другие.

Static Mesh Component [3] - это компонент, который представляет собой статичный объект, состоящий из геометрии и материалов. Он используется для отображения трехмерных объектов в игровом мире. Static Mesh Component имеет ряд преимуществ, таких как более быстрая загрузка и меньшее количество ресурсов, необходимых для отображения объекта. Однако, его использование может привести к проблемам с производительностью, особенно при работе с большим количеством объектов.

Instance Static Mesh [3] - это техника, которая позволяет использовать один и тот же экземпляр Static Mesh Component для отображения нескольких объектов. Это позволяет сократить количество ресурсов, необходимых для отображения объектов, и улучшить производительность игры. Однако, использование Instance Static Mesh может быть ограничено в некоторых случаях, например, когда требуется отображение динамически изменяющихся объектов.

В целом, правильное использование Static Mesh Component и Instance Static Mesh может значительно повлиять на оптимизацию отображения трехмерных элементов в UE4. Однако, выбор метода зависит от конкретных требований проекта и требует тщательного анализа и тестирования.

Оптимизация отрисовки трехмерных объектов в контексте разработки интерактивных элементов пользовательского интерфейса в Unreal Engine 4 является ключевым аспектом для повышения эффективности и производительности игр и приложений. Основной компонент, используемый в этом процессе, - это Static Mesh Component, который позволяет добавлять статические 3D-объекты в игровой мир и контролировать их отображение и поведение.

Одним из методов оптимизации является упрощение геометрии объектов. Использование моделей с меньшим количеством полигонов снижает нагрузку на процессор и видеокарту, что приводит к улучшению производительности. Такой подход

позволяет сохранить визуальное качество при одновременном уменьшении ресурсоемкости.

Другой важный метод оптимизации – это снижение количества отдельных отображаемых объектов на сцене. Важность использования этого метода можно проиллюстрировать следующим примером – в том случае, когда один объект, созданный программистом, состоит из множества разных менее масштабных объектов, которые представляют собой объекты разных классов, их совместная отрисовка очень сильно усложняется как с точки зрения производительности, так и с точки зрения решений по редактированию этих объектов и компонентов. Оптимальным решением в данной ситуации будет создать один «родительский» объект класса, а остальные «дочерние» объекты интерпретировать как компоненты этого объекта, что позволяет упростить создание, дальнейшее изменение и редактирование данного объекта, а также существенно снижает нагрузку на вычислительные элементы компьютера.

Также важно использование текстурных атласов, которые позволяют объединять несколько текстур в одну. Это снижает количество вызовов к видеопамяти и ускоряет процесс отрисовки. Unreal Engine 4 обеспечивает возможности для создания и эффективного использования текстурных атласов, что способствует оптимизации работы с текстурами.

В целом, использование Static Mesh Component в Unreal Engine 4 позволяет проводить эффективную оптимизацию отрисовки трехмерных объектов. Это включает в себя не только технические аспекты, но и учет визуального качества и пользовательского опыта. Правильное использование данного компонента способствует созданию плавной и реалистичной игровой среды, улучшая общее впечатление от игры.

Кроме того, оптимизация в Unreal Engine 4 включает в себя работу с освещением и тенями. Эффективное использование источников света и теней может значительно повышать реалистичность сцены, одновременно снижая нагрузку на систему. Методы, такие как динамическое освещение и отложенное рендеринг, обеспечивают гибкость и производительность при создании сложных освещенных сцен.

Наконец, важным аспектом оптимизации является профилирование и мониторинг производительности. Unreal Engine 4 предоставляет мощные инструменты для

мониторинга и анализа производительности, позволяя разработчикам точно определять узкие места и оптимизировать их. Профилирование помогает в выявлении проблем с производительностью на ранних этапах разработки, что способствует более эффективному и целенаправленному процессу оптимизации.

3. Преимущества использования Static Mesh Component

Static Mesh Component – это мощный инструмент, который позволяет разработчикам создавать сложные сцены, объединяя множество отдельных объектов в единое целое. Этот компонент является одним из ключевых элементов Unreal Engine 4 и предоставляет разработчикам множество возможностей для оптимизации производительности и упрощения процесса создания контента. Одной из главных причин использования Static Mesh Component является возможность быстрого и простого создания сложных сцен, которые могут содержать большое количество объектов. Вместо того чтобы создавать каждый объект отдельно и добавлять его на сцену, можно создать один Static Mesh и использовать его для представления всей группы объектов. Это позволяет существенно сократить время, затрачиваемое на создание и управление объектами, а также упростить процесс оптимизации производительности.

Кроме того, Static Mesh Component может использовать различные оптимизации, такие как отсечение и раннее прекращение отрисовки, чтобы уменьшить количество работы, которую нужно выполнить при отрисовке сцены. Это позволяет значительно повысить производительность при работе со сложными сценами и обеспечить плавную и реалистичную визуализацию. В целом, использование Static Mesh Component является более эффективным способом работы со сложными сценами в Unreal Engine 4. Он позволяет сократить время, затрачиваемое на создание и управление объектами, а также повысить производительность при работе со сложными сценами. Благодаря этому компоненту разработчики могут создавать более качественные игры и приложения, которые работают быстро и плавно даже на ограниченных ресурсах.

Static Mesh Component также предоставляет возможность для более точного контроля над визуальными эффектами и освещением объектов. Разработчики могут настроить различные параметры, такие как текстуры, материалы и освещение, для каждого объекта в сцене. Это позволяет создавать более реалистичные и привлекательные визуальные эффекты, которые улучшают пользовательский опыт и делают игру или приложение более привлекательными для игроков.

Также Static Mesh Component [4] обеспечивает более эффективное использование ресурсов системы. Вместо того чтобы обрабатывать каждый объект отдельно, система

может обрабатывать только один Static Mesh, что позволяет сократить нагрузку на процессор и графический процессор. Это особенно важно при работе с большими сценами или в условиях ограниченной аппаратной поддержки.

4. Структура проекта

Все файлы с исходным кодом расположены в папке «Source», файлы конфигурации – в папке «Config», основной контент проекта – в папке «Content».

Далее изображена блок-схема дерева файлов с исходным кодом (рис. 3):

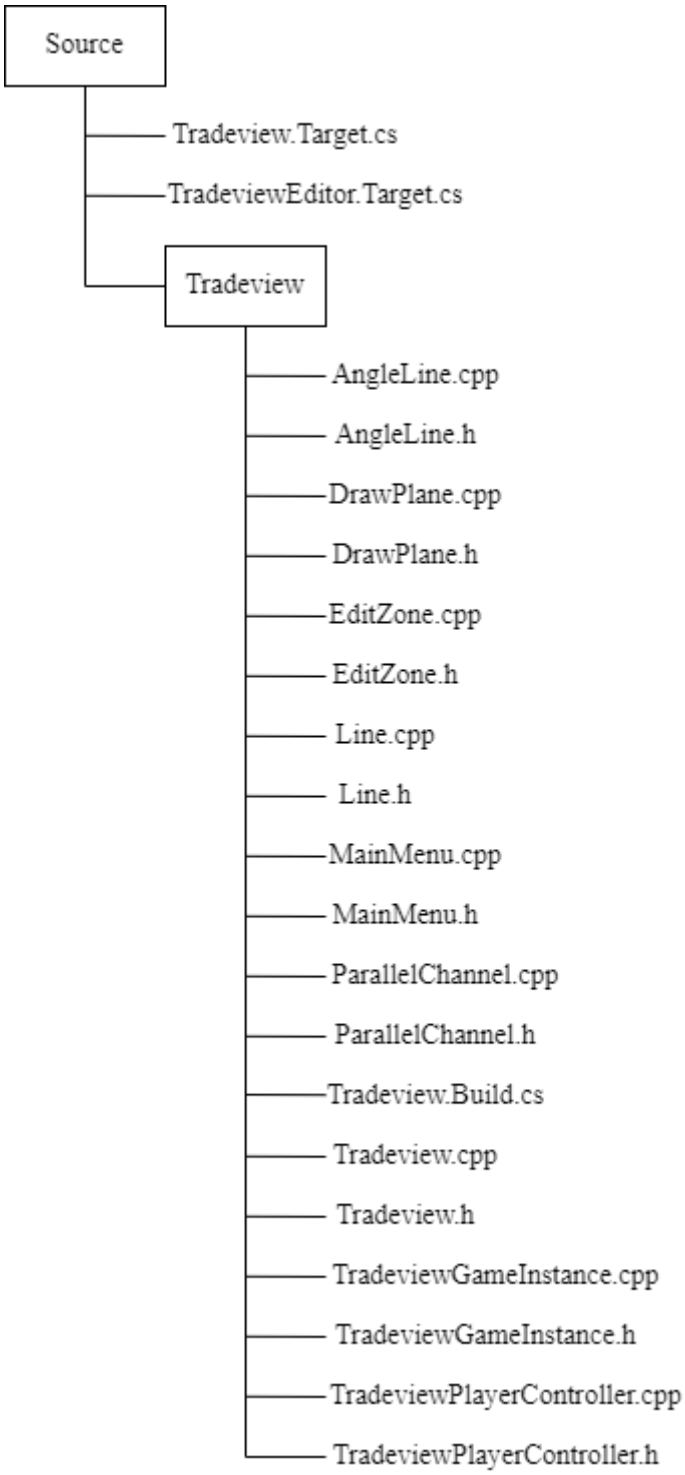

Рисунок 3. Структура папки Source

5. Описание классов

В данном проекте были реализованы следующие классы:

1. AAngleLine – базовый класс, наследуемый от класса AActor, являет собой линию с произвольным углом наклона – динию тренда.

   Листинг 1 – файл AngleLine.h

```
1. UCLASS()
2. class TRADEVIEW_API AAngleLine : public AActor
3. {
4.   GENERATED_BODY()
5.
6. private:
7.   AAngleLine();
8.
9. public:
10.     UPROPERTY(VisibleAnywhere)
11.       UStaticMeshComponent* AngleLine;
12.     UPROPERTY(VisibleAnywhere)
13.       UStaticMeshComponent* DummyMesh;
14.     UPROPERTY(VisibleAnywhere)
15.       UStaticMeshComponent* EditZone1;
16.     UPROPERTY(VisibleAnywhere)
17.       UStaticMeshComponent* EditZone2;
18.     UPROPERTY(VisibleAnywhere)
19.       TArray<USceneComponent*> EditZoneArray;
20.
21.     FVector FirstPointAngle;
22.     FVector SecondPointAngle;
23.
24.     void CreateAngle(FTransform);
25.
26.     UFUNCTION()
27.       void ChangeCursorInTouchZone(UPrimitiveComponent*
   TouchedComponent);
28.     UFUNCTION()
29.       void ChangeCursorToDefault(UPrimitiveComponent*
   TouchedComponent);
30.
31.     virtual void Tick(float DeltaSeconds) override;
32.     virtual void BeginPlay() override;
33.   };
```

2. ALine – базовый класс, наследуемый от класса AActor, являет собой основной элемент отображения функциональных элементов – линию. Отображение элементов происходит после выбора необходимого элемента по нажатию кнопки на виджете, реализованном в классе UMainMenu, и клика в произвольном месте по объекту класса ADrawPlane, отвечающего за плоскость для размещения объектов.

   Листинг 2 – файл Line.h

```
1. UCLASS()
2. class TRADEVIEW_API ALine : public AActor
```

```
3. {
4.   GENERATED_BODY()
5.
6. private:
7.   ALine();
8.
9. protected:
10.     virtual void BeginPlay() override;
11.
12.   public:
13.     virtual void Tick(float DeltaSeconds) override;
14.
15.     UPROPERTY(VisibleAnywhere)
16.       UStaticMeshComponent* Line;
17.     UPROPERTY(VisibleAnywhere)
18.       UStaticMeshComponent* DummyMesh;
19.
20.     UPROPERTY(VisibleAnywhere)
21.       TArray<FVector> PointPosition;
22.
23.     UPROPERTY(VisibleAnywhere)
24.       bool Angle;
25.
26.     UPROPERTY(VisibleAnywhere)
27.       FVector StartTrace;
28.
29.     UFUNCTION()
30.       void ChangeCursorInTouchZone(UPrimitiveComponent*
    TouchedComponent);
31.     UFUNCTION()
32.       void ChangeCursorToDefault(UPrimitiveComponent*
    TouchedComponent);
33.     void CreateHorizontal(FVector);
34.     void CreateVertical(FVector);
35.   };
```

3. ADrawPlane – класс, наследуемый от класса AActor, необходим для размещения на его плоскости функциональных элементов.

Листинг 3 – файл DrawPlane.h

```
1. UCLASS()
2. class TRADEVIEW_API ADrawPlane : public AActor
3. {
4.   GENERATED_BODY()
5.
6. private:
7.   ADrawPlane();
8.
9. protected:
10.   virtual void BeginPlay() override;
11.
12.   public:
13.     virtual void Tick(float DeltaTime) override;
14.
15.     UPROPERTY(VisibleAnywhere)
16.       UStaticMeshComponent* DrawPlane;
17.   };
```

4. AEditZone – класс, наследуемый от класса AActor, являет собой зону редактирования размещенных объектов.

   Листинг 4 – файл EditZone.h

```
UCLASS()
class TRADEVIEW_API AEditZone : public AActor
{
  GENERATED_BODY()

private:
  AEditZone();

protected:
    virtual void BeginPlay() override;

  public:
    virtual void Tick(float DeltaTime) override;

    UPROPERTY(VisibleAnywhere)
      UStaticMeshComponent* EditZone;
  };
```

5. UMainMenu – класс, наследуемый от класса UUserWidget, необходимый для отображения и функционирования виджетов на экране. С помощью него реализованы кнопки перехода к меню выбора размещаемого элемента (кнопка Lines) и непосредственного выбора размещаемого элемента из предложенных вариантов (предложенные варианты: Horizontal – горизонтальная линия, Vertical – вертикальная линия, Angle – “линия тренда”, или линия с произвольным углом наклона, задаваемым пользователем, Channel – параллельный канал, являющий собой три параллельные линии и зону внутри канала).

   Листинг 5 – файл UMainMenu.h

```
UCLASS()
class TRADEVIEW_API UMainMenu : public UUserWidget
{
  GENERATED_BODY()

private:

  UPROPERTY(meta = (BindWidget))
    class UButton* Lines_Button;
    UPROPERTY(meta = (BindWidget))
      class UButton* Horizontal_Button;
    UPROPERTY(meta = (BindWidget))
      class UButton* Vertical_Button;
    UPROPERTY(meta = (BindWidget))
      class UButton* Angle_Button;
    UPROPERTY(meta = (BindWidget))
      class UButton* Channel_Button;

```

```
19.     UPROPERTY(meta = (BindWidget))
20.       class UWidgetSwitcher* MenuSwitch;
21.
22.     UPROPERTY(meta = (BindWidget))
23.       class UWidget* LineButtons;
24.
25.     UPROPERTY(meta = (BindWidget))
26.       class UWidget* BlankScreen;
27.
28.     UFUNCTION()
29.       void OpenLinesMenu();
30.     UFUNCTION()
31.       void HorizontalLine();
32.     UFUNCTION()
33.       void VerticalLine();
34.     UFUNCTION()
35.       void AngleLine();
36.     UFUNCTION()
37.       void ChannelLine();
38.
39.   protected:
40.     virtual bool Initialize();
41.
42.     UPROPERTY(VisibleAnywhere)
43.       class APlayerController* PlayerController;
44.
45.     UPROPERTY(VisibleAnywhere)
46.       class ATradeviewPlayerController* PC;
47.
48.     bool LMOpen = false;
49.
50.   public:
51.     TSubclassOf<UUserWidget> LinesMenuClass;
52.   };
```

6. AParallelChannel – класс, наследуемый от класса AActor, являет собой параллельный канал, ограниченный тремя параллельными линиями, которые перемещаются друг относительно друга только по вертикали, и тач-зоной, обозначающей зону покрытия канала.

Листинг 6 – файл ParallelChannel.h

```
1. UCLASS()
2. class TRADEVIEW_API AParallelChannel : public AActor
3. {
4.   GENERATED_BODY()
5.
6. private:
7.   AParallelChannel();
8.
9. protected:
10.     virtual void BeginPlay() override;
11.
12.   public:
13.     virtual void Tick(float DeltaTime) override;
14.
15.     UPROPERTY(EditAnywhere)
16.       UMaterialInterface* DummyMaterial;
17.     UPROPERTY(EditAnywhere)
18.       UMaterialInterface* MLMaterial;
```

```
19.     UPROPERTY(EditAnywhere)
20.       UMaterialInterface* LMaterial;
21.     UPROPERTY(VisibleAnywhere)
22.       UMaterialInterface* EMaterial;
23.
24.     UPROPERTY(VisibleAnywhere)
25.       UStaticMeshComponent* Channel;
26.     UPROPERTY(VisibleAnywhere)
27.       UStaticMeshComponent* BCLine;
28.     UPROPERTY(VisibleAnywhere)
29.       UStaticMeshComponent* MCLine;
30.     UPROPERTY(VisibleAnywhere)
31.       UStaticMeshComponent* TCLine;
32.
33.     UPROPERTY(VisibleAnywhere)
34.       UStaticMeshComponent* EditZone0;
35.     UPROPERTY(VisibleAnywhere)
36.       UStaticMeshComponent* EditZone1;
37.     UPROPERTY(VisibleAnywhere)
38.       UStaticMeshComponent* EditZone2;
39.     UPROPERTY(VisibleAnywhere)
40.       UStaticMeshComponent* EditZone3;
41.     UPROPERTY(VisibleAnywhere)
42.       UStaticMeshComponent* EditZone4;
43.     UPROPERTY(VisibleAnywhere)
44.       UStaticMeshComponent* EditZone5;
45.
46.     UPROPERTY(VisibleAnywhere)
47.       UProceduralMeshComponent* ChannelDummy;
48.
49.     TArray<FVector> Vertices;
50.     TArray<int32> Triangles;
51.     TArray<FVector2D> UVs;
52.
53.     UPROPERTY(VisibleAnywhere)
54.       FVector FirstPoint;
55.     UPROPERTY(VisibleAnywhere)
56.       FVector SecondPoint;
57.     UPROPERTY(VisibleAnywhere)
58.       FVector ThirdPoint;
59.     UPROPERTY(VisibleAnywhere)
60.       FVector ForthPoint;
61.
62.     UPROPERTY(VisibleAnywhere)
63.       TArray<USceneComponent*> EditZoneArray;
64.
65.     void CreateParallelChannel(ALine*, ALine*);
66.
67.     UFUNCTION()
68.       void ChangeCursorInTouchZone(UPrimitiveComponent*
   TouchedComponent);
69.
70.     UFUNCTION()
71.       void ChangeCursorToDefault(UPrimitiveComponent*
   TouchedComponent);
72.   };
```

7. UTradeviewGameInstance – класс, наследуемый от класса UGameInstance, являющий собой необходимые настройки и положения элементов при запуске и обработке работы проекта. В нем реализовано начальное размещение и

отображение виджета класса UMainMenu, необходимого для работы выбора размещаемого элемента.

Листинг 7 – файл TradeviewGameInstance.h

```
1. UCLASS()
2. class TRADEVIEW_API UTradeviewGameInstance : public UGameInstance
3. {
4.   GENERATED_BODY()
5.
6. private:
7.   UTradeviewGameInstance();
8.
9. public:
10.     UFUNCTION(BlueprintCallable)
11.       void LoadMainMenu();
12.
13.     TSubclassOf<UUserWidget> MenuClass;
14.   };
```

8. ATradeviewPlayerController – класс, наследуемый от класса APlayerController, являет собой основной элемент управления объектов на экране и их логикой. В нем реализовано размещение функциональных элементов в соответствии с произвольными нажатиями пользователя, их изменение и поведение во время изменения и перемещения.

Листинг 8 – файл TradeviewPlayerController.h

```
1. UCLASS()
2. class TRADEVIEW_API ATradeviewPlayerController : public
   APlayerController
3. {
4.   GENERATED_BODY()
5.
6. private:
7.   ATradeviewPlayerController();
8.
9. protected:
10.     void SetupInputComponent();
11.
12.   public:
13.     UPROPERTY(VisibleAnywhere)
14.       AAngleLine* AngleLine;
15.     UPROPERTY(VisibleAnywhere)
16.       FVector FirstPointAngle;
17.     UPROPERTY(VisibleAnywhere)
18.       FVector SecondPointAngle;
19.
20.     UPROPERTY(VisibleAnywhere)
21.       AParallelChannel* Channel;
22.     UPROPERTY(VisibleAnywhere)
23.       ALine* BottomChannelLine;
24.     UPROPERTY(VisibleAnywhere)
25.       ALine* TopChannelLine;
26.     UPROPERTY(VisibleAnywhere)
27.       FVector FirstPointChannel;
28.     UPROPERTY(VisibleAnywhere)
29.       FVector SecondPointChannel;
```

```
30.     UPROPERTY(VisibleAnywhere)
31.       FVector ThirdPointChannel;
32.
33.     UPROPERTY(VisibleAnywhere)
34.       FTransform LineTransform;
35.
36.     UPROPERTY(VisibleAnywhere)
37.       UPrimitiveComponent* GrabbedComponent;
38.     UPROPERTY(VisibleAnywhere)
39.       FVector RelativeGrabLocation;
40.     UPROPERTY(VisibleAnywhere)
41.       float OriginalDistance;
42.     UPROPERTY(VisibleAnywhere)
43.       FVector AncorPoint;
44.     UPROPERTY(VisibleAnywhere)
45.       TArray<USceneComponent*> EditZoneArray;
46.
47.     UPROPERTY(VisibleAnywhere)
48.       float Height;
49.     UPROPERTY(VisibleAnywhere)
50.       bool LineEditing = false;
51.     UPROPERTY(VisibleAnywhere)
52.       bool ChannelEditing = false;
53.     UPROPERTY(VisibleAnywhere)
54.       int ChannelLine = 0;
55.     UPROPERTY(VisibleAnywhere)
56.       bool HeightEditing = false;
57.
58.
59.
60.     virtual void Tick(float DeltaSeconds) override;
61.
62.     void OnMouseClick();
63.     void OnMouseRelease();
64.
65.     void SetHorizontal();
66.     void SetVertical();
67.     void SetAngle();
68.     void SetChannel();
69.
70.     bool GrabMode = true;
71.     bool Horizontal = false;
72.     bool Vertical = false;
73.     bool Angle = false;
74.     bool PlaceAngle = false;
75.     bool ParallelChannel = false;
76.     bool PlaceParallelChannel = false;
77.     bool PlaceParallelTop = false;
78.   };
```

5. Реализация размещения и поведения классов

В меню выбора размещаемых элементов есть 4 опции:

Кнопка Horizontal – отвечает за задание логики размещения горизонтальной линии без возможности ее дальнейшего редактирования, но оставляя возможность ее перемещения по плоскости отображения объектов. При нажатии на кнопку Horizontal запускается логика задания размеров будущей горизонтальной линии и программа переключается в режим ожидания нажатия пользователем в произвольном месте на плоскости отображения. После регистрации нажатия программа получает координаты нажатия на плоскость, после чего создает объект класса Aline, задает ему координаты размещения на плоскости для размещения объектов, затем создает компонент DummyMeshComponent, реализующий тач-зону размещенной горизонтальной линии и присоединяет этот объект к уже размещенной горизонтальной линии. Далее размещенный объект имеет возможность быть перемещенным по вертикали посредством нажатия на его тач-зону и перетягивания его в произвольную точку на рабочей плоскости.

Кнопка Vertical – отвечает за задание логики размещения вертикальной линии без возможности ее дальнейшего редактирования, но оставляя возможность ее перемещения по плоскости отображения объектов. При нажатии на кнопку Vertical запускается логика задания размеров будущей вертикальной линии и программа переключается в режим ожидания нажатия пользователем в произвольном месте на плоскости отображения. После регистрации нажатия программа получает координаты нажатия на плоскость, после чего создает объект класса Aline, задает ему координаты размещения на плоскости для размещения объектов, затем создает компонент DummyMeshComponent, реализующий тач-зону размещенной вертикальной линии и присоединяет этот объект к уже размещенной вертикальной линии. Далее размещенный объект имеет возможность быть перемещенным по горизонтали посредством нажатия на его тач-зону и перетягивания его в произвольную точку на рабочей плоскости.

Кнопка Angle – отвечает за задание логики размещения линии с произвольным углом наклона с возможностью ее дальнейшего перемещения и редактирования. После нажатия кнопки Angle программа ожидает, пока пользователь нажмет в произвольной точке рабочей плоскости, после чего происходит определение координат точки касания

и эти координаты записываются в поле первой точки построения линии (FirstPointAngle). Далее программа на каждый тик считывает положение курсора указателя мыши пользователя и обновляет положение линии так, чтобы концами этой линии являлись уже известные координаты первой точки и проекция положения курсора указателя мыши на рабочую плоскость. Когда пользователь хочет завершить построение линии, ему необходимо еще раз нажать левую кнопку мыши в произвольном месте на рабочей плоскости для считывания координат положения курсора указателя мыши и записи соответствующих координат в поле второй точки построения линии (SecondPointAngle). Построение линии с произвольным углом наклона завершается и строится линия, концы которой имеют координаты, записанные в поля первой и второй точек построения линии, после чего создаются вспомогательные элементы: компонент DummyMeshComponent, являющий собой тач-зону построенной линии с произвольным углом наклона, который после создания присоединяется к уже построенной линии; Два объекта класса AEditZone, являющих собой визуальные поля, обозначающие поля редактирования построенной линии с произвольным углом наклона, которые располагаются на концах построенной линии.

Перемещение линии с произвольным углом наклона осуществляется посредством наведения курсора указателя мыши на тач-зону этой линии, нажатия левой кнопкой мыши по тач-зоне и перетягивания ее в произвольную точку на рабочей плоскости.

Редактирование линии с произвольным углом наклона осуществляется посредством наведения курсора указателя мыши на тач-зону редактируемой линии, после которого на концах линии станут видны поля для редактирования. Для редактирования необходимо переместить курсор в область поля для редактирования, зажать левую кнопку мыши и перетащить поле для редактирования в необходимую пользователю точку.

При редактировании после нажатия левой кнопки мыши в области поля для редактирования линия перестраивается следующим образом:

После нажатия левой кнопкой мыши по одному из двух полей для редактирования координаты противоположного поля для редактирования (то есть соответствующие координаты конца линии) записываются в поле якорной точки (AncorPoint) и остаются неизменными до конца редактирования.

Линия перестраивается относительно двух точек: якорной точки и положения курсора указателя мыши пользователя с необходимыми изменениями положения и масштаба.

После того, как пользователь отпустит левую кнопку мыши, линия выйдет из режима редактирования и будет построена по двум точкам: точке с координатами якорной точки и точки, в которой была отпущена левая кнопка мыши. Положение вспомогательных объектов (тач-зоны и полей для редактирования) изменится соответственно изменениям линии с произвольным углом наклона.

Кнопка Channel – отвечает за задание логики размещения объекта класса AParallelChannel, являющим собой параллельный канал, с возможностью его дальнейшего перемещения и редактирования. После нажатия кнопки Channel программа ожидает, пока пользователь нажмет в произвольной точке рабочей плоскости, после чего происходит определение координат точки касания и эти координаты записываются в поле первой точки построения параллельного канала (FirstPointChannel). Далее программа на каждый тик считывает положение курсора указателя мыши пользователя и обновляет положение первой линии канала так, чтобы концами этой линии являлись уже известные координаты первой точки и проекция положения курсора указателя мыши на рабочую плоскость. Когда пользователь хочет завершить построение первой линии канала, ему необходимо еще раз нажать левую кнопку мыши в произвольном месте на рабочей плоскости для считывания координат положения курсора указателя мыши и записи соответствующих координат в поле второй точки построения первой линии канала (SecondPointChannel). Построение первой линии канала завершается и строится линия, концы которой имеют координаты, записанные в поля первой и второй точек построения канала, после чего создается вторая линия канала, имеющая идентичные вектора положения, вращения и масштаба. Вторая линия канала перемещается строго по вертикали относительно первой линии канала, для чего необходимо было ввести следующие ограничения перемещения второй линии канала:

При перемещении курсора указателя мыши по вертикали вверх и вниз вторая линия канала перемещается на такое же расстояние вверх и вниз соответственно.

При перемещении курсора указателя мыши по горизонтали вправо и влево вторая линия канала перемещается так, чтобы прямая, содержащая вторую линию канала, следовала за курсором указателя мыши.

Координаты положения второй линии канала определяются из отношения разницы координат концов первой линии канала по вертикали и разницы координат концов первой линии канала по горизонтали, умноженного на перемещение курсора указателя мыши по горизонтали.

После нажатия левой кнопки мыши устанавливается положение второй линии канала, после чего строится параллельный канала следующим образом:

В функцию построения параллельного канала передаются обе линии параллельного канала.

Определяются ключевые (крайние) точки параллельного канала: первая и вторая точки параллельного канала, расположенные на концах первой линии канала, определяются из ранее полученных при построении первой линии канала точек и записываются в соответствующие поля; Третья и четвертая точки параллельного канала определяются как сумма второй точки параллельного канала и разницы координат по вертикали линий параллельного канала и сумма первой точки параллельного канала и разницы координат по вертикали линий параллельного канала соответственно.

Строится зона покрытия канала, она же тач-зона канала – процедурно генерируемый меш из четырех треугольных секций, которые задаются с учетом расположения нормалей секций процедурно генерируемого меша: две секции – лицевая сторона – должны иметь нормали, направленные в сторону пользователя, оставшиеся две – тыльная сторона процедурно генерируемого меша – соответственно, в противоположную сторону, от пользователя. Это достигается путем грамотного задания топологии построения треугольных секций процедурно генерируемого меша – по ”правилу правой руки”, то есть при обходе вершин так, как они были расположены при задании топологии процедурно генерируемого меша, необходимо направить пальцы правой руки через вершины треугольной секции и большой палец правой руки, отведенный под 90 градусов, будет показывать направление нормали треугольной секции процедурно генерируемого меша. Это продемонстрировано цифрами красного и синего цвета (рис. 4).

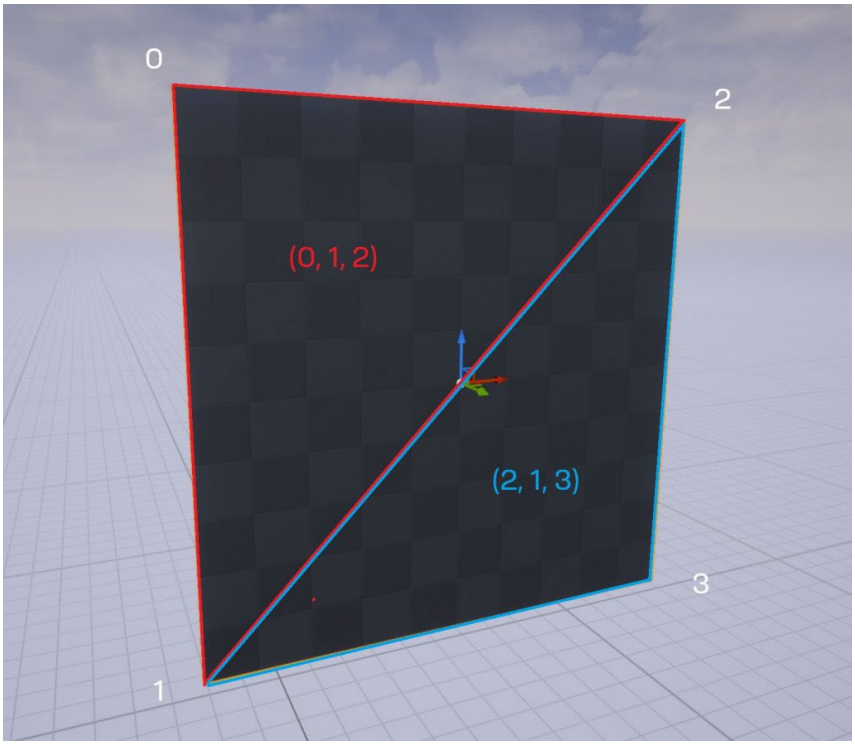

Рисунок 4. Пример построения процедурно генерируемого меша [6]

Создаются и добавляются к параллельному каналу 6 объектов класса AEditZone, являющих собой области полей для редактирования параллельного канала. Они располагаются соответственно концам обеих линий, образующих параллельный канал, по 2 области на каждую линию, и по одной области полей редактирования на центр каждой линии параллельного канала.

После построения параллельного канала пользователь может перемещать и редактировать параллельный канал по своему усмотрению.

Перемещение параллельного канала осуществляется посредством наведения курсора указателя мыши на тач-зону параллельного канала, нажатия левой кнопкой мыши по тач-зоне и перетягивания ее в произвольную точку на рабочей плоскости.

Редактирование параллельного канала осуществляется посредством наведения курсора указателя мыши на тач-зону редактируемого параллельного канала, после которого станут видны все поля для редактирования. Для редактирования необходимо переместить курсор в область поля для редактирования, зажать левую кнопку мыши и перетащить поле для редактирования в необходимую пользователю точку.

При редактировании после нажатия левой кнопки мыши в области поля для редактирования параллельный канал перестраивается следующими образами:

Если курсор указателя мыши находится в области одного из полей редактирования, расположенных на концах линий параллельного канала, то по нажатию левой кнопки мыши в области поля редактирования на одной из линий параллельного канала противоположный конец этой линии становится “якорной точкой” и остается неизменной до конца редактирования.

Линия параллельного канала перестраивается относительно двух точек: якорной точки и положения курсора указателя мыши пользователя с необходимыми изменениями положения и масштаба.

Вторая линия параллельного канала повторяет изменения и перемещения редактируемой линии параллельного канала, но сохраняет параллельность и разницу между линиями параллельного канала по вертикали.

Соответственно перестраивается и тач-зона параллельного канала, сохраняя свою топологию.

Если курсор указателя мыши находится в области одного из полей редактирования, расположенных на центрах линий параллельного канала, то по нажатию левой кнопки мыши в области поля редактирования на одной из линий параллельного канала пользователь может переместить редактируемую линию вертикально вверх или вниз, по следующим принципам:

При перемещении курсора указателя мыши по вертикали вверх и вниз редактируемая линия канала перемещается на такое же расстояние вверх и вниз соответственно.

При перемещении курсора указателя мыши по горизонтали вправо и влево редактируемая линия канала перемещается так, чтобы прямая, содержащая редактируемую линию канала, следовала за курсором указателя мыши.

Координаты положения редактируемой линии канала определяются из отношения разницы координат концов линии канала по вертикали и разницы координат концов линии канала по горизонтали, умноженного на перемещение курсора указателя мыши по горизонтали.

Тач-зона параллельного канала перестраивается соответствующе изменениям параллельного канала.

После отпускания левой кнопки мыши программа выйдет из режима редактирования.

ЗАКЛЮЧЕНИЕ

В данной работе были рассмотрены методы разработки интерактивных элементов пользовательского интерфейса в трехмерном движке Unreal Engine 4, с акцентом на использование Static Mesh Component. Этот компонент представляет собой мощный инструмент для создания статичных 3D-моделей в игровом мире, обладающий широким спектром возможностей для настройки и адаптации под конкретные нужды проекта.

Static Mesh Component позволяет разработчикам создавать разнообразные интерактивные элементы пользовательского интерфейса, такие как кнопки, переключатели и слайдеры, с высокой степенью настраиваемости в плане формы, размера, текстур и анимаций. Это обеспечивает гибкость и уникальность в создании пользовательских интерфейсов, что способствует созданию более привлекательного и интуитивно понятного опыта для пользователей.

Кроме того, Static Mesh Component взаимодействует с другими компонентами и функциями Unreal Engine 4, позволяя разработчикам интегрировать различные виды интерактивности и эффектов. Такая функциональность расширяет границы традиционных пользовательских интерфейсов и позволяет создавать более сложные и динамичные взаимодействия в игровом мире.

Также была подчеркнута важность оптимизации при работе с трехмерными объектами, которая является ключевым фактором для обеспечения плавной работы и высокой производительности игр. Static Mesh Component поддерживает различные оптимизации, включая упрощение геометрии, использование текстурных атласов и оптимизацию использования вычислительных мощностей компьютера, что позволяет создавать эффективные и быстро загружаемые интерактивные элементы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Unreal Engine 4 Documentation // Unreal Engine Documentation URL: https://docs.unrealengine.com/. Дата обращения: 11.10.2023.
2. Display aspect ratio // Wikipedia, the free encyclopedia URL: https://en.wikipedia.org/wiki/Display_aspect_ratio/. Дата обращения: 09.10.2023.
3. Geometry instancing // Wikipedia, the free encyclopedia URL: https://en.wikipedia.org/wiki/Geometry_instancing. Дата обращения: 13.11.2023.
4. Working with Static Mesh Actors and Components // Epic Games Development Documentation URL: https://dev.epicgames.com/community/learning/tutorials/qzoY/unreal-engine-working-with-static-mesh-actors-and-components. Дата обращения: 17.11.2023.
5. Tools referencing // Tradingview site URL: https://ru.tradingview.com/ Дата обращения: 18.11.2023.
6. Procedural Mesh Component Documentation // Unreal Engine Documentation URL: https://docs.unrealengine.com/4.27/en-US/API/Plugins/ProceduralMeshComponent/UProceduralMeshComponent/. Дата обращения: 23.11.2023