|  |  |
| --- | --- |
|  | **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)***\_

**ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

Студент\_\_\_\_\_\_\_\_\_\_\_Афиногенов Михаил Алексеевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*фамилия, имя, отчество*

Группа\_\_***РК6-81Б***\_\_\_

Тип практики \_\_\_\_\_\_\_\_\_\_\_***Преддипломная***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_Афиногенов М.А.\_**

*подпись, дата фамилия, и.о.*

Руководитель практики

от кафедры **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_Витюков Ф.А.\_\_\_\_**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***2024 г.***

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой *РК6*

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ *А.П. Карпенко* \_

« \_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

**ЗАДАНИЕ**

**на прохождение производственной практики**

**\_\_\_\_\_\_*Преддипломная*\_\_\_\_\_\_**

Тип практики

Студент

Афиногенов Михаил Алексеевич\_\_\_\_\_\_\_\_\_\_\_ \_\_***4***\_\_ курса группы \_***РК6-81Б***\_

Фамилия Имя Отчество № курса индекс группы

в период с \_***13 мая 2024***\_\_\_ г. по ***26 мая 2024*** г.

*Предприятие:* \_\_\_***НИИ АПП МГТУ им. Н.Э. Баумана***\_\_\_\_\_\_\_

*Подразделение:* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(отдел/сектор/цех)

*Руководитель практики от предприятия (наставник):*

***Киселев Игорь Алексеевич, директор НИИ АПП МГТУ им. Н.Э. Баумана***

(Фамилия Имя Отчество полностью, должность)

*Руководитель практики от кафедры:*

Витюков Федор Андреевич, ст. преподаватель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия Имя Отчество полностью, должность)

*Задание:*

**1.** *Провести исследование способов сохранения, реализуемых в Unreal Engine 4*

**2.** *Разработать метод сохранения объектов на сцене*

**3.** *Разработать метод загрузки объектов на сцену после сохранения*

Дата выдачи задания ***14 мая 2024***г.

Руководитель практики от предприятия  **\_\_\_\_\_\_\_\_\_\_ /\_\_*И.А. Киселев*\_\_/**

Руководитель практики от кафедры  **\_\_\_\_\_\_\_\_\_\_ /\_\_Ф.А. Витюков\_/**

Студент **\_\_\_\_\_\_\_\_\_\_ /\_М.А.Афиногенов/**

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 4](#_Toc167390711)

[**1.** **ПОСТАНОВКА ЗАДАЧИ** 5](#_Toc167390712)

[**2.** **КРАТКИЙ ОТЧЕТ О ВЫПОЛНЕННЫХ РАБОТАХ** 6](#_Toc167390713)

[**ЗАКЛЮЧЕНИЕ** 12](#_Toc167390714)

[**СПИСОК ЛИТЕРАТУРЫ** 13](#_Toc167390715)

**ВВЕДЕНИЕ**

Существует несколько способов сохранения объектов в Unreal Engine 4. Один из них - использование системы сохранения уровня, которая позволяет сохранять все объекты и настройки уровня в один файл. Другой способ - использование системы сохранения игрока, которая сохраняет только информацию о конкретном игроке, его инвентаре, положении и прогрессе в игре. Также можно использовать собственные методы сохранения, создавая собственную систему сохранения данных объектов. Каждый из этих способов имеет свои преимущества и недостатки, и выбор подходящего зависит от конкретных потребностей и целей разработки проекта.

USaveGame - это класс в Unreal Engine 4, который используется для сохранения и загрузки игровых данных. Этот класс позволяет разработчикам сохранять состояние игры, такие как прогресс игрока, настройки игры, инвентарь и многое другое, чтобы игрок мог продолжить игру с того же места, где он остановился.

USaveGame - это базовый класс, который позволяет создавать пользовательские классы для сохранения данных игры. Он предназначен для хранения информации, которая должна быть сохранена между игровыми сессиями, чтобы игрок мог вернуться к игре позже и продолжить с сохраненным прогрессом.

1. **ПОСТАНОВКА ЗАДАЧИ**

В ходе выполнения преддипломной практики необходимо:

1. Изучить способы и методы сохранения ранее размещенных на сцене объектов и выбрать оптимальный из них.
2. Разработать методы сохранения трехмерных функциональных элементов на сцене: горизонтальная и вертикальная линии, линия тренда, параллельный канал.
3. Разработать методы загрузки сохраненных трехмерных функциональных элементов и размещения их на сцене.

1. **КРАТКИЙ ОТЧЕТ О ВЫПОЛНЕННЫХ РАБОТАХ**

В ходе данной работы был разработан специальный класс UMySaveGame, наследуемый от базового класса USaveGame, необходимого для сохранения состояния игры, объектов и информации об объектах на сцене.

Для сохранения данных с помощью USaveGame можно использовать функции SaveGameToSlot и LoadGameFromSlot, которые позволяют сохранить данные в файл и загрузить их обратно из файла. Также можно использовать другие методы, такие как SaveGameToMemory и LoadGameFromMemory, для сохранения данных в память и загрузки их оттуда.

Для того, чтобы сохранить функциональные элементы на сцене, был разработан следующий функционал класса UMySaveGame:

1. Структура FSaveLine – необходима для сохранения горизонтальных и вертикальных линий, содержит в себе поля Position – положение линии на сцене, и Horizontal – булевая переменная, определяющая какой будет размещенная линия – горизонтальной и вертикальной. Для размещения линий на сцене используются уже разработанные методы класса ALine – CreateHorizontal и CreateVertical.
2. Структура FSaveAngleLine – необходима для сохранения линий тренда – линий с произвольным углом наклона, - содержит в себе поля FirstPoint – положение первой точки линии тренда (одного конца отрезка), SecondPoint – положение второй точки линии тренда (другого конца отрезка), LineTransform – структура, содержащая вектор вращения, вектор положения и вектор масштаба объекта. Для размещения линий тренда на сцене используется уже разработанный метод класса AAngleLine – CreateAngle.
3. Структура FSaveChannel – необходима для сохранения параллельных каналов, содержит в себе поля FirstPoint – положение первой точки параллельного канала, SecondPoint – положение второй точки параллельного канала, ThirdPoint – положение третьей точки параллельного канала, ForthPoint – положение четвертой точки параллельного канала, BotLine – объект класса ALine, необходимый для сохранения нижней линии параллельного канала, TopLine – объект класса ALine, необходимый для сохранения верхней линии параллельного канала. Для размещения параллельного канала на сцене используется уже разработанный метод класса AParallelChannel – CreateChannel.

Листинг 1 – файл MySaveGame.h

|  |
| --- |
| 1. USTRUCT() 2. struct FSaveLine 3. { 4. GENERATED\_BODY() 5. public: 6. UPROPERTY(EditAnywhere) 7. FVector Position; 8. UPROPERTY(EditAnywhere) 9. bool Horizontal; 10. }; 11. USTRUCT() 12. struct FSaveAngleLine 13. { 14. GENERATED\_BODY() 15. public: 16. UPROPERTY(EditAnywhere) 17. FVector FirstPoint; 18. UPROPERTY(EditAnywhere) 19. FVector SecondPoint; 20. UPROPERTY(EditAnywhere) 21. FTransform LineTransform; 22. }; 23. USTRUCT() 24. struct FSaveChannel 25. { 26. GENERATED\_BODY() 27. public: 28. UPROPERTY(EditAnywhere) 29. FVector FirstPoint; 30. UPROPERTY(EditAnywhere) 31. FVector SecondPoint; 32. UPROPERTY(EditAnywhere) 33. FVector ThirdPoint; 34. UPROPERTY(EditAnywhere) 35. FVector ForthPoint; 36. UPROPERTY(EditAnywhere) 37. ALine\* BotLine; 38. UPROPERTY(EditAnywhere) 39. ALine\* TopLine; 40. }; 41. UCLASS() 42. class TRADEVIEW\_API UMySaveGame : public USaveGame 43. { 44. GENERATED\_BODY() 45. public: 46. UPROPERTY(EditAnywhere) 47. TArray<FSaveLine> SaveLines; 48. UPROPERTY(EditAnywhere) 49. TArray<FSaveAngleLine> SaveAngleLines; 50. UPROPERTY(EditAnywhere) 51. TArray<FSaveChannel> SaveChannel; 52. UMySaveGame(); 53. }; |

Сохранение объектов на сцене происходит следующим образом:

1. Создается объект класса UMySaveGame, благодаря которому осуществимо сохранение расположенных на сцене объектов.
2. Проводится поиск по объектам класса ALine, который описывает горизонтальные и вертикальные линии.
3. Создается подходящая структура, входящая в состав объекта класса UMySaveGame, в поля которой поочередно записываются все необходимые для сохранения и дальнейшей загрузки данные.
4. Полученная структура сохраняется в поле объекта класса UMySaveGame.
5. п.п 2-4 повторяются для следующих элементов: линий тренда (класс AnglrLine) и параллельных каналов (класс AParallelChannel).
6. Полученный на выходе объект класса UMySaveGame сохраняется в специальный слот, отведенный автоматически Unreal Engine 4, и создается файл, описывающий сохраненный элементы.

Листинг 2 – файл TradeviewPlayerController.cpp – функция SaveGame

|  |
| --- |
| 1. void ATradeviewPlayerController::SaveGame(FString FileName) 2. { 3. int Index; 4. if (FileName == "Gazprom\_13-14") Index = 0; 5. else if (FileName == "KMAZ\_13-14") Index = 1; 6. else Index = 2; 7. GI = GetGameInstance(); 8. UMySaveGame\* SavedData = Cast<UMySaveGame>(UGameplayStatics::CreateSaveGameObject(UMySaveGame::StaticClass())); 9. TArray<AActor\*> Out; 10. UGameplayStatics::GetAllActorsOfClass(GetWorld(), ALine::StaticClass(), Out); 11. for (AActor\* FActor : Out) 12. { 13. FSaveLine FSL; 14. FSL.Position = FActor->GetActorLocation(); 15. FSL.Horizontal = Cast<ALine>(FActor)->Horizontal; 16. SavedData->SaveLines.Add(FSL); 17. } 18. Out.Empty(); 19. UGameplayStatics::GetAllActorsOfClass(GetWorld(), AAngleLine::StaticClass(), Out); 20. for (AActor\* FActor : Out) 21. { 22. FSaveAngleLine FSAL; 23. FSAL.FirstPoint = Cast<AAngleLine>(FActor)->FirstPointAngle; 24. FSAL.SecondPoint = Cast<AAngleLine>(FActor)->SecondPointAngle; 25. FSAL.LineTransform = FActor->GetActorTransform(); 26. SavedData->SaveAngleLines.Add(FSAL); 27. } 28. Out.Empty(); 29. UGameplayStatics::GetAllActorsOfClass(GetWorld(), AParallelChannel::StaticClass(), Out); 30. for (AActor\* FActor : Out) 31. { 32. FSaveChannel FSC; 33. FSC.FirstPoint = Cast<AParallelChannel>(FActor)->FirstPoint; 34. FSC.SecondPoint = Cast<AParallelChannel>(FActor)->SecondPoint; 35. FSC.ThirdPoint = Cast<AParallelChannel>(FActor)->ThirdPoint; 36. FSC.ForthPoint = Cast<AParallelChannel>(FActor)->ForthPoint; 37. FSC.BotLine = Cast<AParallelChannel>(FActor)->BCLineSave; 38. FSC.TopLine = Cast<AParallelChannel>(FActor)->TCLineSave; 39. SavedData->SaveChannel.Add(FSC); 40. } 41. Out.Empty(); 42. UGameplayStatics::SaveGameToSlot(SavedData, FileName, 0 /\*Index\*/); 43. } |

Перед загрузкой и размещением сохраненных объектов на сцену производится очистка сцены.

Листинг 3 – файл TradeviewPlayerController.cpp – функция ClearActors

|  |
| --- |
| 1. void ATradeviewPlayerController::ClearActors() 2. { 3. TArray<AActor\*> Out; 4. UGameplayStatics::GetAllActorsOfClass(GetWorld(), ALine::StaticClass(), Out); 5. for (AActor\* ActorFound : Out) { ActorFound->Destroy(); } 6. Out.Empty(); 7. UGameplayStatics::GetAllActorsOfClass(GetWorld(), AAngleLine::StaticClass(), Out); 8. for (AActor\* ActorFound : Out) { ActorFound->Destroy(); } 9. Out.Empty(); 10. UGameplayStatics::GetAllActorsOfClass(GetWorld(), AParallelChannel::StaticClass(), Out); 11. for (AActor\* ActorFound : Out) { ActorFound->Destroy(); } 12. Out.Empty(); 13. } |

Загрузка и размещение объектов на сцене происходит следующим образом:

1. Создается объект класса UMySaveGame, необходимый для загрузки и расшифровки сохраненных ранее данных.
2. В полученном объекте класса поочередно расшифровываются структуры данных, содержащие сохраненные элементы.
3. В зависимости от того, с какой структурой данных проводится работа в данный момент, вызываются соответствующие функции отрисовки функциональных элементов.

Листинг 4 – файл TradeviewPlayerController.cpp – функция LoadGame

|  |
| --- |
| 1. void ATradeviewPlayerController::LoadGame(FString FileName) 2. { 3. int Index; 4. if (FileName == "Gazprom\_13-14") Index = 0; 5. else if (FileName == "KMAZ\_13-14") Index = 1; 6. else Index = 2; 7. if (UGameplayStatics::DoesSaveGameExist(FileName, 0 /\*Index\*/)) 8. { 9. GI = GetGameInstance(); 10. UMySaveGame\* LoadedData = Cast<UMySaveGame>(UGameplayStatics::LoadGameFromSlot(FileName, 0 /\*Index\*/)); 11. if (LoadedData->SaveLines.Num() > 0) 12. for (FSaveLine ActorToSpawn : LoadedData->SaveLines) 13. { 14. ALine\* Line = GetWorld()->SpawnActor<ALine>(); 15. ActorToSpawn.Horizontal ? Line->CreateHorizontal(ActorToSpawn.Position) : Line->CreateVertical(ActorToSpawn.Position); 16. } 17. if (LoadedData->SaveAngleLines.Num() > 0) 18. for (FSaveAngleLine ActorToSpawn : LoadedData->SaveAngleLines) 19. { 20. AAngleLine\* Line = GetWorld()->SpawnActor<AAngleLine>(); 21. Line->CreateAngle(ActorToSpawn.LineTransform, ActorToSpawn.FirstPoint, ActorToSpawn.SecondPoint); 22. } 23. if (LoadedData->SaveChannel.Num() > 0) 24. for (FSaveChannel ActorToSpawn : LoadedData->SaveChannel) 25. { 26. AParallelChannel\* LoadChannel = GetWorld()->SpawnActor<AParallelChannel>(); 27. LoadChannel->CreateParallelChannel(ActorToSpawn.BotLine, ActorToSpawn.TopLine, ActorToSpawn.FirstPoint, ActorToSpawn.SecondPoint, ActorToSpawn.ThirdPoint, ActorToSpawn.ForthPoint); 28. } 29. } 30. } |

**ЗАКЛЮЧЕНИЕ**

В ходе данной практической работы был исследован класс USaveGame в Unreal Engine 4, который отвечает за сохранение и загрузку игровых данных. Были рассмотрены его структура, функционал и применение. USaveGame представляет собой важный компонент любой игры, построенной на движке Unreal Engine 4, так как он позволяет сохранять и восстанавливать состояние игры, что улучшает игровой процесс и повышает удобство использования. Кроме того, этот класс предоставляет разработчикам гибкие возможности для управления сохранениями и загрузками, что позволяет создавать индивидуальные решения для каждого конкретного проекта. В рамках преддипломной практики были изучены основные методы класса USaveGame, такие как SaveGame, LoadGame, а также рассмотрены способы их использования в коде. Было продемонстрировано, как создать собственный класс сохранений, наследующийся от USaveGame, и реализовать в нем необходимый функционал.

**СПИСОК ЛИТЕРАТУРЫ**

1. USaveGame base class // Unreal Documentation URL: <https://docs.unrealengine.com/4.27/en-US/API/Runtime/Engine/GameFramework/USaveGame/> . Дата обращения: 16.05.2024
2. USTRUCT base class // Unreal Documentation URL: <https://docs.unrealengine.com/4.26/en-US/API/Runtime/CoreUObject/UObject/UStruct/> . Дата обращения: 19.05.2024
3. *UGameplayStatics class in Unreal Engine // VREALMATIC Site URL:* [*https://vrealmatic.com/unreal-engine/classes/ugameplaystatics*](https://vrealmatic.com/unreal-engine/classes/ugameplaystatics) *. Дата обращения: 18.05.2024*