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.

З А Д А Н И Е
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

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.

2. 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.
6. public:
7. UPROPERTY(EditAnywhere)
8. FVector Position;
9.
10.   UPROPERTY(EditAnywhere)
11.   bool Horizontal;
12.   };
13.
14.   USTRUCT()
15.   struct FSaveAngleLine
16.   {
17.   GENERATED_BODY()
18.
19.   public:
20.   UPROPERTY(EditAnywhere)
21.   FVector FirstPoint;
22.
23.   UPROPERTY(EditAnywhere)
24.   FVector SecondPoint;
25.
26.   UPROPERTY(EditAnywhere)
27.   FTransform LineTransform;
28.   };
29.
30.   USTRUCT()
31.   struct FSaveChannel
32.   {
33.   GENERATED_BODY()
34.
35.   public:
36.   UPROPERTY(EditAnywhere)
37.   FVector FirstPoint;
38.   UPROPERTY(EditAnywhere)
39.   FVector SecondPoint;
40.   UPROPERTY(EditAnywhere)
41.   FVector ThirdPoint;
42.   UPROPERTY(EditAnywhere)
43.   FVector ForthPoint;
44.
45.   UPROPERTY(EditAnywhere)
46.   ALine* BotLine;
47.   UPROPERTY(EditAnywhere)
48.   ALine* TopLine;
49.   };
```

```
50.
51.   UCLASS()
52.   class TRADEVIEW_API UMySaveGame : public USaveGame
53.   {
54.   GENERATED_BODY()
55.
56.   public:
57.   UPROPERTY(EditAnywhere)
58.   TArray<FSaveLine> SaveLines;
59.
60.   UPROPERTY(EditAnywhere)
61.   TArray<FSaveAngleLine> SaveAngleLines;
62.
63.   UPROPERTY(EditAnywhere)
64.   TArray<FSaveChannel> SaveChannel;
65.
66.   UMySaveGame();
67.   };
```

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.
8. GI = GetGameInstance();
```

```
9. UMySaveGame* SavedData =
   Cast<UMySaveGame>(UGameplayStatics::CreateSaveGameObject(UMySaveGame:
   :StaticClass()));

10.   TArray<AActor*> Out;
11.
12.   UGameplayStatics::GetAllActorsOfClass(GetWorld(),
   ALine::StaticClass(), Out);
13.   for (AActor* FActor : Out)
14.   {
15.   FSaveLine FSL;
16.   FSL.Position = FActor->GetActorLocation();
17.   FSL.Horizontal = Cast<ALine>(FActor)->Horizontal;
18.   SavedData->SaveLines.Add(FSL);
19.   }
20.   Out.Empty();
21.
22.   UGameplayStatics::GetAllActorsOfClass(GetWorld(),
   AAngleLine::StaticClass(), Out);
23.   for (AActor* FActor : Out)
24.   {
25.   FSaveAngleLine FSAL;
26.   FSAL.FirstPoint = Cast<AAngleLine>(FActor)->FirstPointAngle;
27.   FSAL.SecondPoint = Cast<AAngleLine>(FActor)->SecondPointAngle;
28.   FSAL.LineTransform = FActor->GetActorTransform();
29.   SavedData->SaveAngleLines.Add(FSAL);
30.   }
31.   Out.Empty();
32.
33.   UGameplayStatics::GetAllActorsOfClass(GetWorld(),
   AParallelChannel::StaticClass(), Out);
34.   for (AActor* FActor : Out)
35.   {
36.   FSaveChannel FSC;
37.   FSC.FirstPoint = Cast<AParallelChannel>(FActor)->FirstPoint;
38.   FSC.SecondPoint = Cast<AParallelChannel>(FActor)->SecondPoint;
39.   FSC.ThirdPoint = Cast<AParallelChannel>(FActor)->ThirdPoint;
40.   FSC.ForthPoint = Cast<AParallelChannel>(FActor)->ForthPoint;
41.   FSC.BotLine = Cast<AParallelChannel>(FActor)->BCLineSave;
42.   FSC.TopLine = Cast<AParallelChannel>(FActor)->TCLineSave;
43.   SavedData->SaveChannel.Add(FSC);
44.   }
45.   Out.Empty();

46.   UGameplayStatics::SaveGameToSlot(SavedData, FileName, 0
   /*Index*/);
47. }
```

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.
```

```
5. UGameplayStatics::GetAllActorsOfClass(GetWorld(),
   ALine::StaticClass(), Out);
6. for (AActor* ActorFound : Out) { ActorFound->Destroy(); }
7. Out.Empty();
8. UGameplayStatics::GetAllActorsOfClass(GetWorld(),
   AAngleLine::StaticClass(), Out);
9. for (AActor* ActorFound : Out) { ActorFound->Destroy(); }
10.   Out.Empty();
11.
12.   UGameplayStatics::GetAllActorsOfClass(GetWorld(),
   AParallelChannel::StaticClass(), Out);
13.   for (AActor* ActorFound : Out) { ActorFound->Destroy(); }
14.   Out.Empty();
15. }
```

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.
8. if (UGameplayStatics::DoesSaveGameExist(FileName, 0 /*Index*/))
9. {
10.   GI = GetGameInstance();
11.   UMySaveGame* LoadedData =
   Cast<UMySaveGame>(UGameplayStatics::LoadGameFromSlot(FileName, 0
   /*Index*/));
12.
13.   if (LoadedData->SaveLines.Num() > 0)
14.   for (FSaveLine ActorToSpawn : LoadedData->SaveLines)
15.   {
16.   ALine* Line = GetWorld()->SpawnActor<ALine>();
17.   ActorToSpawn.Horizontal ? Line-
   >CreateHorizontal(ActorToSpawn.Position) : Line-
   >CreateVertical(ActorToSpawn.Position);
18.   }
19.   if (LoadedData->SaveAngleLines.Num() > 0)
20.   for (FSaveAngleLine ActorToSpawn : LoadedData->SaveAngleLines)
```

```
21.    {
22.    AAngleLine* Line = GetWorld()->SpawnActor<AAngleLine>();
23.    Line->CreateAngle(ActorToSpawn.LineTransform,
       ActorToSpawn.FirstPoint, ActorToSpawn.SecondPoint);
24.    }
25.    if (LoadedData->SaveChannel.Num() > 0)
26.    for (FSaveChannel ActorToSpawn : LoadedData->SaveChannel)
27.    {
28.    AParallelChannel* LoadChannel = GetWorld()-
       >SpawnActor<AParallelChannel>();
29.    LoadChannel->CreateParallelChannel(ActorToSpawn.BotLine,
       ActorToSpawn.TopLine, ActorToSpawn.FirstPoint,
       ActorToSpawn.SecondPoint, ActorToSpawn.ThirdPoint,
       ActorToSpawn.ForthPoint);
30.    }
31.    }
32.    }
```

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 . Дата обращения: 18.05.2024*