ДОДАТОК А

СЛАЙДИ ПРЕЗЕНТАЦІЇ


Міністерство освіти і науки України
Харківський національний університет радіоелектроніки
АТЕСТАЦІЙНА РОБОТА МАГІСТРА
Дослідження алгоритмів стабілізації та прицілювання при стрільбі з танка, балістики траєкторії стрільби на симуляторі віртуальної реальності.
Керівник проекту
доц. каф. ПІ
Назаров О.С.
Виконав
Студент групи ІПЗМ-18-3
Артюхов О.Д.
1


Мета роботи
Об'єкт розробки – Розробка танкового Симулятору у вигляді ігрового додатка до віртуальної реальності
Мета розробки – розробка алгоритмів балістики траєкторії стрільби до симулятору віртуальної реальності
Метод рішення – Windows, HTC vive, мова скриптингу Blueprint.
2

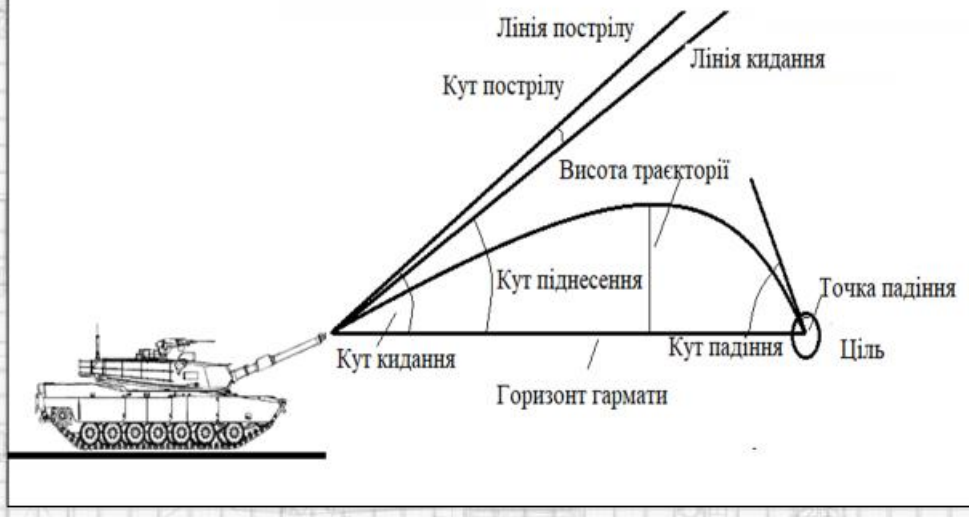
Постановка задачі

- Проаналізувати предметну область
- дослідження сил впливають на тіло під час його руху
- Розробити повноцінне програмне забезпечення
- Протестувати розроблене програмне забезпечення

3

Балістика

Траєкторія польоту



4

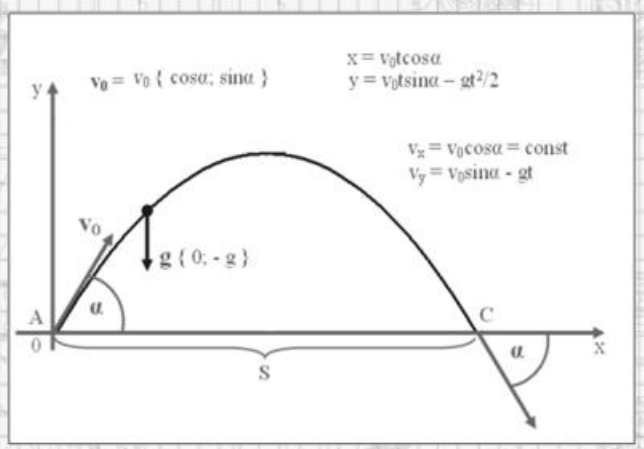
Фактори, які впливають на траєкторію польоту:

- Початкова швидкість і кут
- Форма і геометрія снаряда
- Опору середовища
- інші фізичні параметри, такі як вологість, тиск, або вплив вітру, турбулентність і т.д

5

Постановка класичної завдання

Рух тіла, кинутого під кутом до горизонту



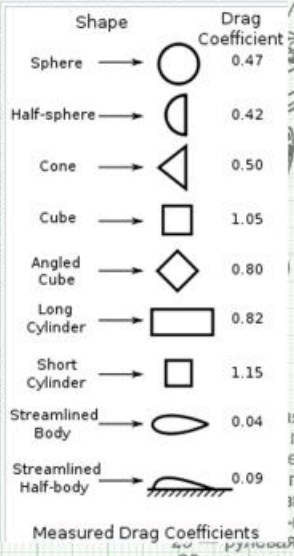
6

коефіцієнт лобового опору

Коефіцієнт опору визначається як

$$C_f = \frac{2F}{pv^2S}$$

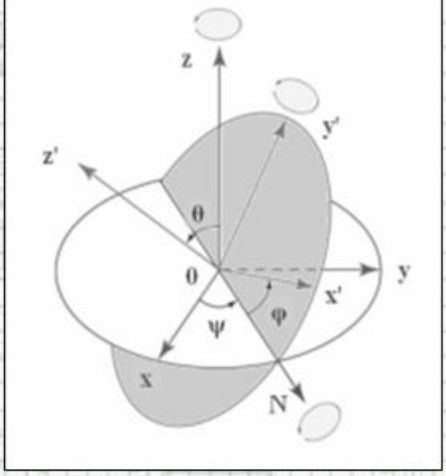
F - це сила опору,
p- щільність середовища,
v - це швидкість потоку об'єкта по відношенню до рідини,
S - характерна площа перпендикулярно до потоку .



7

6 ступенів свободи

Кути Ейлера



8

Обчислення центру інерції снаряда

З геометричної моделі снаряда можна висловити:

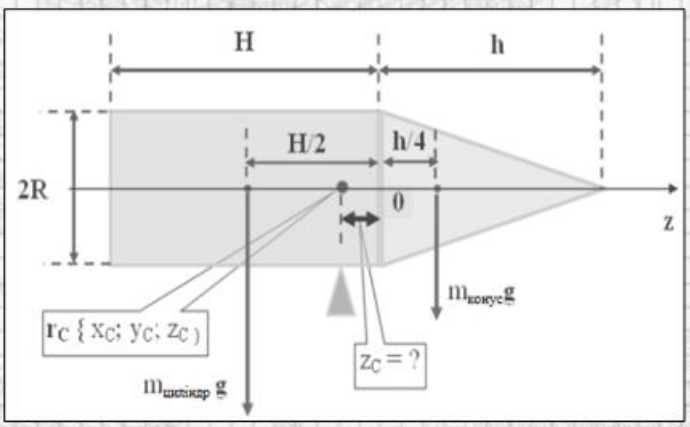
$$m_{цилиндр} \cdot g \cdot \left(\frac{H}{2} - z_C \right)$$

$$m_{конус} \cdot g \cdot \left(\frac{h}{4} + z_C \right)$$

$$m_{цилиндр} = \pi R^2 H \rho \qquad m_{конус} = \pi R^2 h \rho / 3$$

$$z_C = \frac{1}{4} \cdot \frac{6H^2 - h^2}{3H + h}$$

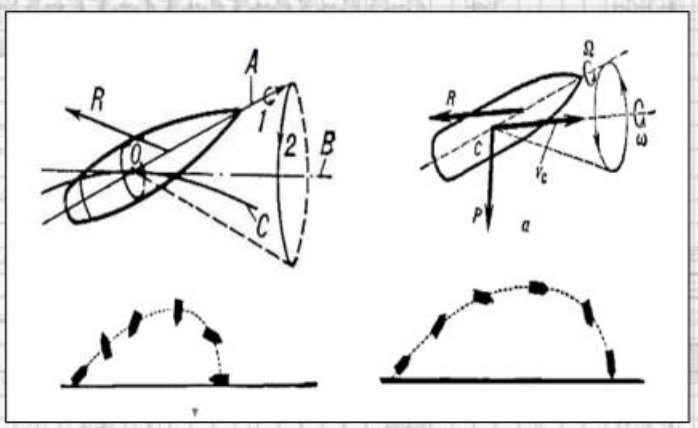
Геометрична модель снаряда



9

Обертання навколо нерухомої осі

Геометрична модель руху снаряда



10

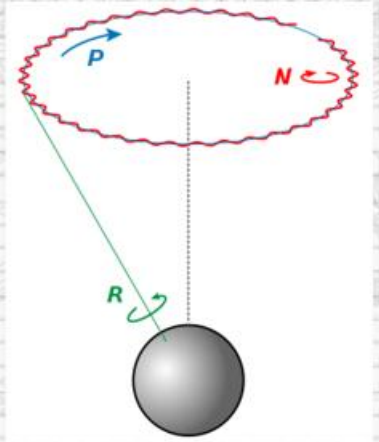
Обертання навколо нерухомої осі

Геометрична модель обертання снаряда

обертання (R, зелений)

прецесія (P, синій)

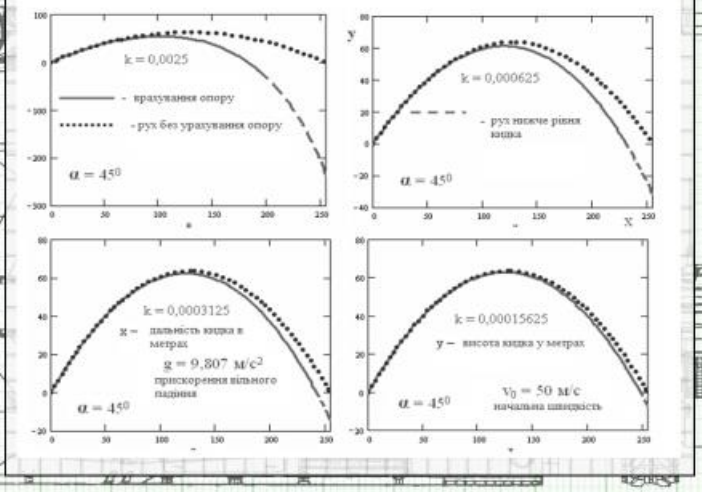
нутація (N, червоний)



11

Опір середовища

Залежність руху тіла від опору середовища



12

Відстань в 3х мірному просторі

13

Відстань в 3х мірному просторі

діагональ зеленого(нижнього) прямокутника

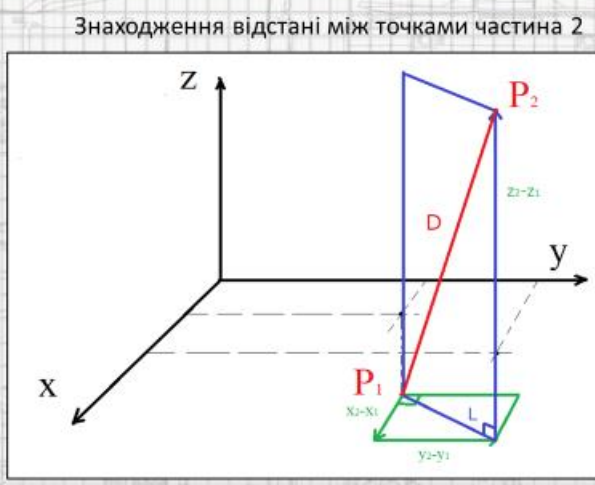
$$L^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2$$

діагональ синього(верхнього) прямокутника

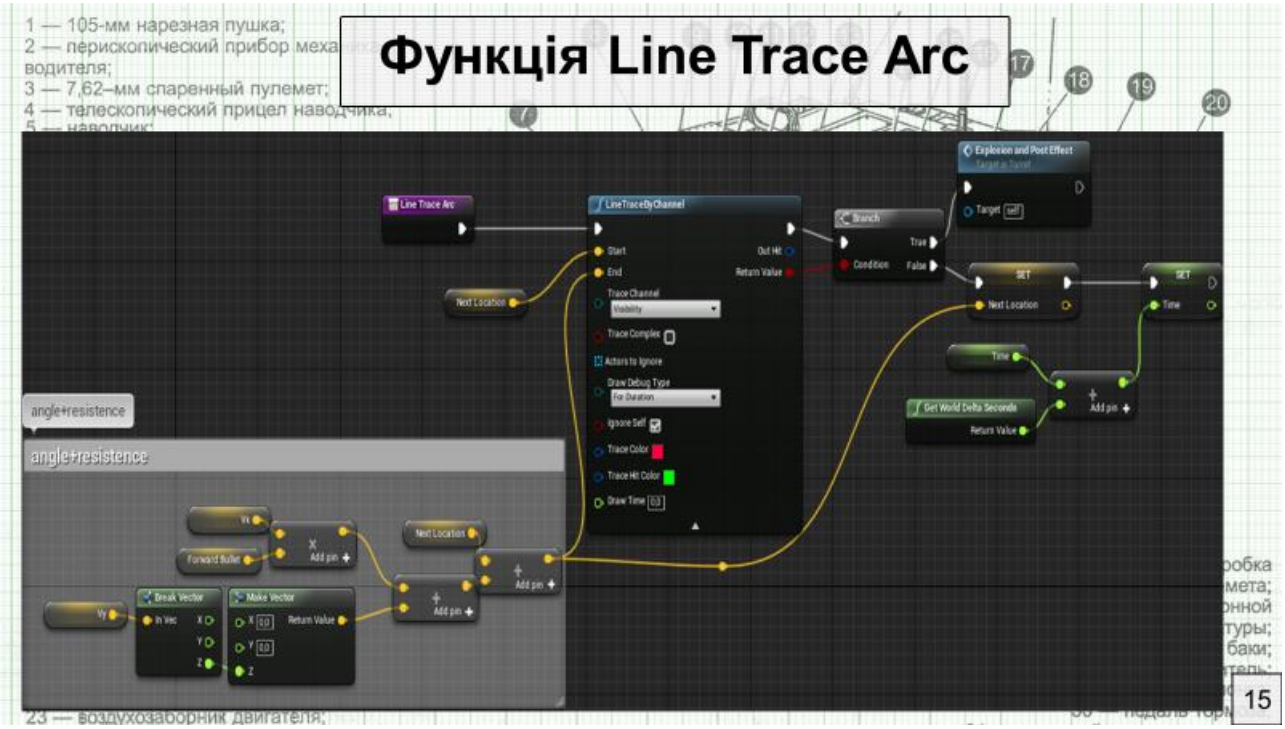
$$D^2 = L^2 + (z_2 - z_1)^2$$

Висловимо кінцеву формулу для знаходження дистанції:

$$d(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$



14


Функція Line Trace Arc
Line Trace Arc
LineTraceByChannel
Start
End
Out Hit
Return Value
Trace Channel
Trace Complex
Actors to Ignore
Draw Debug Type
Ignore Self
Trace Color
Trace Hit Color
Draw Time
Branch
Condition
True
False
Explosion and Post Effect
Target
SET
Next Location
Time
Get World Delta Seconds
Add pin
angle+resistence
Forward Bullet
Break Vector
Make Vector
15


Висновки
➢ Проведено аналіз предметної галузі
➢ Були досліджені сили що впливають на тіло під час його руху
➢ Розроблені алгоритми які вирішують поставлені завдання
➢ Розроблен ігровий додаток
➢ Протестував розроблене програмне забезпечення
16


1 — 105-мм нарезная пушка;
2 — перископический прибор механика–водителя;
3 — 7,62–мм спаренный пулемет;
4 — телескопический прицел наводчика;
5 — наводчик;
6 — основной прицел наводчика;
7 — гранатомет для постановки дымовых завес;
8 — пульт управления командира танка;
9 — 12,7–мм пулемет;
10 — основной прицел командира;
11 — прицел командира для стрельбы из пулемета;
12 — командир танка;
13 — 7,62–мм пулемет;
14 — люк;
15 — вспомогательный вентилятор;
16 — анемометр;
17 — отсек для боеприпасов;
18 — газотурбинный двигатель AGT–1500;
19 — воздушный фильтр;
20 — кормовые топливные баки;
21 — основной вентилятор;
22 — трансмиссия;
23 — воздухозаборник двигателя;
25 — патронная коробка спаренного пулемета;
26 — блоки электронной аппаратуры;
27 — передние топливные баки;
28 — механик–водитель;
29 — рулевая кол
30 — педаль тор
Дякую за увагу!
17

ДОДАТОК Б
КОД ІГРОВОГО ДОДАТКУ

```
Aturret_C__pf1690381860::Aturret_C__pf1690381860(const
FObjectInitializer& ObjectInitializer) : Super(ObjectInitializer)
{
   if(HasAnyFlags(RF_ClassDefaultObject)                                  &&
   (Aturret_C__pf1690381860::StaticClass() == GetClass()))
   {

      Aturret_C__pf1690381860::__CustomDynamicClassInitialization(CastChe
   cked<UDynamicClass>(GetClass()));
   }

   bpv__Scene__pf                                                         =
   CreateDefaultSubobject<USceneComponent>(TEXT("Scene"));
   bpv__turret__pf                                                        =
   CreateDefaultSubobject<UStaticMeshComponent>(TEXT("turret"));
   bpv__gun__pf                                                           =
   CreateDefaultSubobject<UStaticMeshComponent>(TEXT("gun"));
   bpv__Camera__pf                                                        =
   CreateDefaultSubobject<UCameraComponent>(TEXT("Camera"));
   bpv__Arrow__pf                                                         =
   CreateDefaultSubobject<UArrowComponent>(TEXT("Arrow"));
   bpv__shoot__pf                                                         =
   CreateDefaultSubobject<UBillboardComponent>(TEXT("shoot"));
   bpv__Forvardhorisont__pf                                               =
   CreateDefaultSubobject<UArrowComponent>(TEXT("Forvardhorisont"));
   bpv__SceneCaptureComponent2D__pf                                       =
   CreateDefaultSubobject<USceneCaptureComponent2D>(TEXT("SceneCaptureCom
   ponent2D"));
   RootComponent = bpv__Scene__pf;
   bpv__Scene__pf->CreationMethod = EComponentCreationMethod::Native;
   static TWeakObjectPtr<UProperty> __Local__1{};
   const UProperty* __Local__0 = __Local__1.Get();
   if (nullptr == __Local__0)
   {
      __Local__0            =            (UActorComponent::StaticClass())-
   >FindPropertyByName(FName(TEXT("bCanEverAffectNavigation")));
      check(__Local__0);
      __Local__1 = __Local__0;
```

```
}
    (((UBoolProperty*)__Local__0)-
    >SetPropertyValue_InContainer((bpv__Scene__pf), false, 0));
    bpv__turret__pf->CreationMethod = EComponentCreationMethod::Native;
    bpv__turret__pf->AttachToComponent(bpv__Scene__pf,
    FAttachmentTransformRules::KeepRelativeTransform );
    auto&      __Local__2      =      (*(AccessPrivateProperty<UStaticMesh*
    >((bpv__turret__pf), UStaticMeshComponent::__PPO__StaticMesh() )));
    __Local__2                                                          =
    CastChecked<UStaticMesh>(CastChecked<UDynamicClass>(Aturret_C__pf16903
    81860::StaticClass())->UsedAssets[0], ECastCheckedType::NullAllowed);
    bpv__turret__pf->OverrideMaterials = TArray<UMaterialInterface*> ();
    bpv__turret__pf->OverrideMaterials.Reserve(1);
    bpv__turret__pf-
    >OverrideMaterials.Add(CastChecked<UMaterialInterface>(CastChecked<UDy
    namicClass>(Aturret_C__pf1690381860::StaticClass())->UsedAssets[1],
    ECastCheckedType::NullAllowed));
    (((UBoolProperty*)__Local__0)-
    >SetPropertyValue_InContainer((bpv__turret__pf), true, 0));
    bpv__gun__pf->CreationMethod = EComponentCreationMethod::Native;
    bpv__gun__pf->AttachToComponent(bpv__turret__pf,
    FAttachmentTransformRules::KeepRelativeTransform );
    auto&      __Local__3      =      (*(AccessPrivateProperty<UStaticMesh*
    >((bpv__gun__pf), UStaticMeshComponent::__PPO__StaticMesh() )));
    __Local__3                                                          =
    CastChecked<UStaticMesh>(CastChecked<UDynamicClass>(Aturret_C__pf16903
    81860::StaticClass())->UsedAssets[2], ECastCheckedType::NullAllowed);
    bpv__gun__pf->OverrideMaterials = TArray<UMaterialInterface*> ();
    bpv__gun__pf->OverrideMaterials.Reserve(1);
    bpv__gun__pf-
    >OverrideMaterials.Add(CastChecked<UMaterialInterface>(CastChecked<UDy
    namicClass>(Aturret_C__pf1690381860::StaticClass())->UsedAssets[1],
    ECastCheckedType::NullAllowed));
    bpv__gun__pf->RelativeLocation    =    FVector(50.000000,    0.000000,
    0.000000);
    (((UBoolProperty*)__Local__0)-
    >SetPropertyValue_InContainer((bpv__gun__pf), true, 0));
    bpv__Camera__pf->CreationMethod = EComponentCreationMethod::Native;
```

```
bpv__Camera__pf->AttachToComponent(bpv__gun__pf,
FAttachmentTransformRules::KeepRelativeTransform );
(((UBoolProperty*)__Local__0)-
>SetPropertyValue_InContainer((bpv__Camera__pf), false, 0));
bpv__Arrow__pf->CreationMethod = EComponentCreationMethod::Native;
bpv__Arrow__pf->AttachToComponent(bpv__gun__pf,
FAttachmentTransformRules::KeepRelativeTransform );
bpv__Arrow__pf->ArrowColor = FColor(79, 255, 143, 255);
bpv__Arrow__pf->RelativeLocation    =    FVector(50.000000,    0.000000,
0.000000);
(((UBoolProperty*)__Local__0)-
>SetPropertyValue_InContainer((bpv__Arrow__pf), false, 0));
bpv__shoot__pf->CreationMethod = EComponentCreationMethod::Native;
bpv__shoot__pf->AttachToComponent(bpv__Arrow__pf,
FAttachmentTransformRules::KeepRelativeTransform );
(((UBoolProperty*)__Local__0)-
>SetPropertyValue_InContainer((bpv__shoot__pf), false, 0));
bpv__shoot__pf->bIsEditorOnly = true;
bpv__Forvardhorisont__pf->CreationMethod                              =
EComponentCreationMethod::Native;
bpv__Forvardhorisont__pf->AttachToComponent(bpv__gun__pf,
FAttachmentTransformRules::KeepRelativeTransform );
bpv__Forvardhorisont__pf->RelativeLocation     =     FVector(50.000000,
0.000000, 0.000000);
(((UBoolProperty*)__Local__0)-
>SetPropertyValue_InContainer((bpv__Forvardhorisont__pf), false, 0));
bpv__SceneCaptureComponent2D__pf->CreationMethod                      =
EComponentCreationMethod::Native;
bpv__SceneCaptureComponent2D__pf->AttachToComponent(bpv__gun__pf,
FAttachmentTransformRules::KeepRelativeTransform );
bpv__SceneCaptureComponent2D__pf->TextureTarget                       =
CastChecked<UTextureRenderTarget2D>(CastChecked<UDynamicClass>(Aturret
_C__pf1690381860::StaticClass())->UsedAssets[3],
ECastCheckedType::NullAllowed);
bpv__SceneCaptureComponent2D__pf->ShowFlagSettings                    =
TArray<FEngineShowFlagsSetting> ();
bpv__SceneCaptureComponent2D__pf-
>ShowFlagSettings.AddUninitialized(1);
```

```
FEngineShowFlagsSetting::StaticStruct()->InitializeStruct(bpv__SceneCaptureComponent2D__pf->ShowFlagSettings.GetData(), 1);
auto& __Local__4 = bpv__SceneCaptureComponent2D__pf->ShowFlagSettings[0];
__Local__4.ShowFlagName = FString(TEXT("TemporalAA"));
__Local__4.Enabled = true;
bpv__SceneCaptureComponent2D__pf->RelativeLocation = FVector(70.000000, 0.000000, 0.000000);
(((UBoolProperty*)__Local__0)->SetPropertyValue_InContainer((bpv__SceneCaptureComponent2D__pf), false, 0));
bpv__ForwardBullet__pf = FVector(0.000000, 0.000000, 0.000000);
bpv__NextLocation__pf = FVector(0.000000, 0.000000, 0.000000);
bpv__BlockHit__pf = true;
bpv__Gravity__pf = 9.820000f;
bpv__Mass__pf = 11.400000f;
bpv__GravityOffSet__pf = FVector(0.000000, 0.000000, 0.000000);
bpv__InitialVelocity__pf = 1700.000000f;
bpv__Time__pf = 1.000000f;
bpv__MaxVertical__pf = 45.000000f;
bpv__MinVertical__pf = -5.000000f;
bpv__HorisontalRotationValue__pf = FRotator(0.000000, 0.000000, 0.000000);
bpv__AxisHorizontalValue__pf = 0.000000f;
bpv__RotationGunValue__pf = 0.000000f;
bpv__AxisVerticalValue__pf = 0.000000f;
bpv__VerticalRotationValue__pf = FRotator(0.000000, 0.000000, 0.000000);
bpv__TimerRatio__pf = 0.010000f;
bpv__Cos__pf = 0.000000f;
bpv__Sin__pf = 0.000000f;
bpv__RatioPower__pf = 1.000000f;
bpv__Cf__pf = 0.000000f;
bpv__Fc__pf = 0.000000f;
bpv__Ax__pf = FVector(0.000000, 0.000000, 0.000000);
bpv__Ay__pf = FVector(0.000000, 0.000000, 0.000000);
bpv__WorldScale__pf = 0.010000f;
```

```
	bpv__VectorT__pf = FVector(0.000000, 0.000000, 0.000000);
	bpv__VectorR__pf = FVector(0.000000, 0.000000, 0.000000);
	bpv__Vx__pf = FVector(0.000000, 0.000000, 0.000000);
	bpv__Vy__pf = FVector(0.000000, 0.000000, 0.000000);
	bpv__V0__pf = 0.000000f;
	bpv__X__pf = FVector(0.000000, 0.000000, 0.000000);
	bpv__Y__pf = FVector(0.000000, 0.000000, 0.000000);
	bpv__V0x__pf = FVector(0.000000, 0.000000, 0.000000);
	bpv__V0y__pf = FVector(0.000000, 0.000000, 0.000000);
	bpv__znak__pf = 0.000000f;
	PrimaryActorTick.bCanEverTick = true;
}
void	Aturret_C__pf1690381860::PostLoadSubobjects(FObjectInstancingGraph*
	OuterInstanceGraph)
{
	Super::PostLoadSubobjects(OuterInstanceGraph);
	if(bpv__Scene__pf)
	{
	   bpv__Scene__pf->CreationMethod = EComponentCreationMethod::Native;
	}
	if(bpv__turret__pf)
	{
	   bpv__turret__pf->CreationMethod = EComponentCreationMethod::Native;
	}
	if(bpv__gun__pf)
	{
	   bpv__gun__pf->CreationMethod = EComponentCreationMethod::Native;
	}
	if(bpv__Camera__pf)
	{
	   bpv__Camera__pf->CreationMethod = EComponentCreationMethod::Native;
	}
	if(bpv__Arrow__pf)
	{
	   bpv__Arrow__pf->CreationMethod = EComponentCreationMethod::Native;
	}
	if(bpv__shoot__pf)
	{
```

```
        bpv__shoot__pf->CreationMethod = EComponentCreationMethod::Native;
    }
    if(bpv__Forvardhorisont__pf)
    {
        bpv__Forvardhorisont__pf->CreationMethod                    =
    EComponentCreationMethod::Native;
    }
    if(bpv__SceneCaptureComponent2D__pf)
    {
        bpv__SceneCaptureComponent2D__pf->CreationMethod            =
    EComponentCreationMethod::Native;
    }
}
void
    Aturret_C__pf1690381860::__CustomDynamicClassInitialization(UDynamicCl
    ass* InDynamicClass)
{
    ensure(0 == InDynamicClass->ReferencedConvertedFields.Num());
    ensure(0 == InDynamicClass->MiscConvertedSubobjects.Num());
    ensure(0 == InDynamicClass->DynamicBindingObjects.Num());
    ensure(0 == InDynamicClass->ComponentTemplates.Num());
    ensure(0 == InDynamicClass->Timelines.Num());
    ensure(nullptr == InDynamicClass->AnimClassImplementation);
    InDynamicClass->AssembleReferenceTokenStream();
    FConvertedBlueprintsDependencies::FillUsedAssetsInDynamicClass(InDynam
    icClass, &__StaticDependencies_DirectlyUsedAssets);
    auto    __Local__5    =    NewObject<USceneComponent>(InDynamicClass,
    USceneComponent::StaticClass(), TEXT("DefaultSceneRoot_GEN_VARIABLE"),
    (EObjectFlags)0x00280029);
    InDynamicClass->ComponentTemplates.Add(__Local__5);
    auto                          __Local__6                            =
    NewObject<UInputActionDelegateBinding>(InDynamicClass,
    UInputActionDelegateBinding::StaticClass(),
    TEXT("InputActionDelegateBinding_1"), (EObjectFlags)0x00000000);
    InDynamicClass->DynamicBindingObjects.Add(__Local__6);
    auto __Local__7 = NewObject<UInputAxisDelegateBinding>(InDynamicClass,
    UInputAxisDelegateBinding::StaticClass(),
    TEXT("InputAxisDelegateBinding_1"), (EObjectFlags)0x00000000);
```

```
    InDynamicClass->DynamicBindingObjects.Add(__Local__7);
    static TWeakObjectPtr<UProperty> __Local__9{};
    const UProperty* __Local__8 = __Local__9.Get();
    if (nullptr == __Local__8)
    {
       __Local__8            =            (UActorComponent::StaticClass())-
    >FindPropertyByName(FName(TEXT("bCanEverAffectNavigation")));
       check(__Local__8);
       __Local__9 = __Local__8;
    }
    (((UBoolProperty*)__Local__8)-
    >SetPropertyValue_InContainer((__Local__5), false, 0));
    __Local__6->InputActionDelegateBindings                                =
    TArray<FBlueprintInputActionDelegateBinding> ();
    __Local__6->InputActionDelegateBindings.AddUninitialized(1);
    FBlueprintInputActionDelegateBinding::StaticStruct()-
    >InitializeStruct(__Local__6->InputActionDelegateBindings.GetData(),
    1);
    auto& __Local__10 = __Local__6->InputActionDelegateBindings[0];
    __Local__10.InputActionName = FName(TEXT("Firegun"));
    __Local__10.FunctionNameToBind                                         =
    FName(TEXT("InpActEvt_Firegun_K2Node_InputActionEvent_0"));
    __Local__7->InputAxisDelegateBindings                                  =
    TArray<FBlueprintInputAxisDelegateBinding> ();
    __Local__7->InputAxisDelegateBindings.AddUninitialized(2);
    FBlueprintInputAxisDelegateBinding::StaticStruct()-
    >InitializeStruct(__Local__7->InputAxisDelegateBindings.GetData(), 2);
    auto& __Local__11 = __Local__7->InputAxisDelegateBindings[0];
    __Local__11.InputAxisName = FName(TEXT("TurretRotateH"));
    __Local__11.FunctionNameToBind                                         =
    FName(TEXT("InpAxisEvt_TurretRotateH_K2Node_InputAxisEvent_0"));
    auto& __Local__12 = __Local__7->InputAxisDelegateBindings[1];
    __Local__12.InputAxisName = FName(TEXT("TurretRotateV"));
    __Local__12.FunctionNameToBind                                         =
    FName(TEXT("InpAxisEvt_TurretRotateV_K2Node_InputAxisEvent_1"));
}
void       Aturret_C__pf1690381860::bpf__ExecuteUbergraph_turret__pf_0(int32
    bpp__EntryPoint__pf)
```

```
{
	FVector
	bpfv__CallFunc_K2_GetComponentLocation_ReturnValue__pf(EForceInit::For
	ceInit);
	float bpfv__CallFunc_Multiply_FloatFloat_ReturnValue__pf{};
	FTimerHandle bpfv__CallFunc_K2_SetTimer_ReturnValue__pf{};
	FVector
	bpfv__CallFunc_GetForwardVector_ReturnValue__pf(EForceInit::ForceInit)
	;
	FRotator
	bpfv__CallFunc_K2_GetComponentRotation_ReturnValue__pf(EForceInit::For
	ceInit);
	bool bpfv__CallFunc_LessEqual_FloatFloat_ReturnValue__pf{};
	float bpfv__CallFunc_Square_ReturnValue__pf{};
	float bpfv__CallFunc_Subtract_FloatFloat_ReturnValue__pf{};
	float bpfv__CallFunc_Sqrt_ReturnValue__pf{};
	FVector
	bpfv__CallFunc_GetForwardVector_ReturnValue1__pf(EForceInit::ForceInit
	);
	float bpfv__CallFunc_Square_ReturnValue1__pf{};
	FVector
	bpfv__CallFunc_GetForwardVector_ReturnValue2__pf(EForceInit::ForceInit
	);
	float bpfv__CallFunc_Square_ReturnValue2__pf{};
	float bpfv__CallFunc_Square_ReturnValue3__pf{};
	float bpfv__CallFunc_Square_ReturnValue4__pf{};
	float bpfv__CallFunc_Add_FloatFloat_ReturnValue2__pf{};
	float bpfv__CallFunc_Multiply_FloatFloat_ReturnValue3__pf{};
	float bpfv__CallFunc_Add_FloatFloat_ReturnValue3__pf{};
	float bpfv__CallFunc_Square_ReturnValue5__pf{};
	float bpfv__CallFunc_Sqrt_ReturnValue1__pf{};
	float bpfv__CallFunc_Multiply_FloatFloat_ReturnValue4__pf{};
	float bpfv__CallFunc_Square_ReturnValue6__pf{};
	float bpfv__CallFunc_Multiply_FloatFloat_ReturnValue5__pf{};
	float bpfv__CallFunc_Add_FloatFloat_ReturnValue4__pf{};
	float bpfv__CallFunc_Add_FloatFloat_ReturnValue5__pf{};
	float bpfv__CallFunc_Add_FloatFloat_ReturnValue6__pf{};
	float bpfv__CallFunc_Add_FloatFloat_ReturnValue7__pf{};
```

```
float bpfv__CallFunc_Sqrt_ReturnValue2__pf{};
float bpfv__CallFunc_Multiply_FloatFloat_ReturnValue6__pf{};
float bpfv__CallFunc_Divide_FloatFloat_ReturnValue__pf{};
int32 __CurrentState = bpp__EntryPoint__pf;
do
{
   switch( __CurrentState )
   {
   case 1:
         {
         }
   case 2:
         {
               bpf__V0xV0y__pf();
         }
   case 3:
         {
               if(::IsValid(bpv__Arrow__pf))
               {
                     bpfv__CallFunc_GetForwardVector_ReturnValue__pf    =
bpv__Arrow__pf->USceneComponent::GetForwardVector();
               }
               bpv__ForwardBullet__pf                                   =
bpfv__CallFunc_GetForwardVector_ReturnValue__pf;
         }
   case 4:
         {
               if(::IsValid(bpv__Arrow__pf))
               {

   bpfv__CallFunc_K2_GetComponentLocation_ReturnValue__pf             =
bpv__Arrow__pf->USceneComponent::K2_GetComponentLocation();
               }
               bpv__NextLocation__pf                                    =
bpfv__CallFunc_K2_GetComponentLocation_ReturnValue__pf;
         }
   case 5:
         {
```

```
            bpf__dragxcoefficient__pfT(/*out*/
b0l__CallFunc_drag_coefficient_Cf__pf);
        }
   case 6:
        {
            bpfv__CallFunc_Multiply_FloatFloat_ReturnValue__pf      =
UKismetMathLibrary::Multiply_FloatFloat(bpv__TimerRatio__pf,
10.000000);
            bpv__RatioPower__pf                                     =
bpfv__CallFunc_Multiply_FloatFloat_ReturnValue__pf;
        }
   case 7:
        {
            bpv__Cf__pf = b0l__CallFunc_drag_coefficient_Cf__pf;
        }
   case 8:
        {
            bpv__Time__pf = 1.000000;
        }
   case 9:
        {
            bpfv__CallFunc_K2_SetTimer_ReturnValue__pf              =
UKismetSystemLibrary::K2_SetTimer(this,  FString(TEXT("LineTraceArc")),
bpv__TimerRatio__pf, true);
        }
   case 10:
        {
            if(::IsValid(bpv__Arrow__pf))
            {

   bpfv__CallFunc_K2_GetComponentRotation_ReturnValue__pf            =
bpv__Arrow__pf->USceneComponent::K2_GetComponentRotation();
            }

   UKismetMathLibrary::BreakRotator(bpfv__CallFunc_K2_GetComponentRota
tion_ReturnValue__pf,   /*out*/   b0l__CallFunc_BreakRotator_Roll1__pf,
/*out*/         b0l__CallFunc_BreakRotator_Pitch1__pf,         /*out*/
b0l__CallFunc_BreakRotator_Yaw1__pf);
```

```
            bpfv__CallFunc_LessEqual_FloatFloat_ReturnValue__pf     =
UKismetMathLibrary::LessEqual_FloatFloat(b0l__CallFunc_BreakRotator_Pi
tch1__pf, 0.000000);
            if (!bpfv__CallFunc_LessEqual_FloatFloat_ReturnValue__pf)
            {
                  __CurrentState = 15;
                  break;
            }
      }
   case 11:
      {
            bpv__znak__pf = 0.000000;
      }
   case 12:
      {
            if(::IsValid(bpv__Forvardhorisont__pf))
            {
                  bpfv__CallFunc_GetForwardVector_ReturnValue1__pf   =
bpv__Forvardhorisont__pf->USceneComponent::GetForwardVector();
            }

   UKismetMathLibrary::BreakVector(bpfv__CallFunc_GetForwardVector_Ret
urnValue1__pf,    /*out*/    b0l__CallFunc_BreakVector_X__pf,    /*out*/
b0l__CallFunc_BreakVector_Y__pf,                                  /*out*/
b0l__CallFunc_BreakVector_Z__pf);
            bpfv__CallFunc_Square_ReturnValue1__pf                  =
UKismetMathLibrary::Square(b0l__CallFunc_BreakVector_Z__pf);
            if(::IsValid(bpv__Arrow__pf))
            {
                  bpfv__CallFunc_GetForwardVector_ReturnValue2__pf   =
bpv__Arrow__pf->USceneComponent::GetForwardVector();
            }
            bpfv__CallFunc_Square_ReturnValue2__pf                  =
UKismetMathLibrary::Square(b0l__CallFunc_BreakVector_Y__pf);

   UKismetMathLibrary::BreakVector(bpfv__CallFunc_GetForwardVector_Ret
urnValue2__pf,    /*out*/    b0l__CallFunc_BreakVector_X1__pf,    /*out*/
```

```
b01__CallFunc_BreakVector_Y1__pf,                                   /*out*/
b01__CallFunc_BreakVector_Z1__pf);
                bpfv__CallFunc_Square_ReturnValue3__pf                  =
UKismetMathLibrary::Square(b01__CallFunc_BreakVector_X__pf);
                bpfv__CallFunc_Square_ReturnValue4__pf                  =
UKismetMathLibrary::Square(b01__CallFunc_BreakVector_Z1__pf);
                bpfv__CallFunc_Add_FloatFloat_ReturnValue2__pf          =
UKismetMathLibrary::Add_FloatFloat(bpfv__CallFunc_Square_ReturnValue3_
_pf, bpfv__CallFunc_Square_ReturnValue2__pf);
                bpfv__CallFunc_Multiply_FloatFloat_ReturnValue3__pf     =
UKismetMathLibrary::Multiply_FloatFloat(b01__CallFunc_BreakVector_Z1__
pf, b01__CallFunc_BreakVector_Z__pf);
                bpfv__CallFunc_Add_FloatFloat_ReturnValue3__pf          =
UKismetMathLibrary::Add_FloatFloat(bpfv__CallFunc_Add_FloatFloat_Retur
nValue2__pf, bpfv__CallFunc_Square_ReturnValue1__pf);
                bpfv__CallFunc_Square_ReturnValue5__pf                  =
UKismetMathLibrary::Square(b01__CallFunc_BreakVector_Y1__pf);
                bpfv__CallFunc_Sqrt_ReturnValue1__pf                    =
UKismetMathLibrary::Sqrt(bpfv__CallFunc_Add_FloatFloat_ReturnValue3__p
f);
                bpfv__CallFunc_Multiply_FloatFloat_ReturnValue4__pf     =
UKismetMathLibrary::Multiply_FloatFloat(b01__CallFunc_BreakVector_Y1__
pf, b01__CallFunc_BreakVector_Y__pf);
                bpfv__CallFunc_Square_ReturnValue6__pf                  =
UKismetMathLibrary::Square(b01__CallFunc_BreakVector_X1__pf);
                bpfv__CallFunc_Multiply_FloatFloat_ReturnValue5__pf     =
UKismetMathLibrary::Multiply_FloatFloat(b01__CallFunc_BreakVector_X1__
pf, b01__CallFunc_BreakVector_X__pf);
                bpfv__CallFunc_Add_FloatFloat_ReturnValue4__pf          =
UKismetMathLibrary::Add_FloatFloat(bpfv__CallFunc_Square_ReturnValue6_
_pf, bpfv__CallFunc_Square_ReturnValue5__pf);
                bpfv__CallFunc_Add_FloatFloat_ReturnValue5__pf          =
UKismetMathLibrary::Add_FloatFloat(bpfv__CallFunc_Multiply_FloatFloat_
ReturnValue5__pf,
bpfv__CallFunc_Multiply_FloatFloat_ReturnValue4__pf);
                bpfv__CallFunc_Add_FloatFloat_ReturnValue6__pf          =
UKismetMathLibrary::Add_FloatFloat(bpfv__CallFunc_Add_FloatFloat_Retur
nValue4__pf, bpfv__CallFunc_Square_ReturnValue4__pf);
```

```
            bpfv__CallFunc_Add_FloatFloat_ReturnValue7__pf          =
UKismetMathLibrary::Add_FloatFloat(bpfv__CallFunc_Add_FloatFloat_Retur
nValue5__pf, bpfv__CallFunc_Multiply_FloatFloat_ReturnValue3__pf);
            bpfv__CallFunc_Sqrt_ReturnValue2__pf                    =
UKismetMathLibrary::Sqrt(bpfv__CallFunc_Add_FloatFloat_ReturnValue6__p
f);
            bpfv__CallFunc_Multiply_FloatFloat_ReturnValue6__pf     =
UKismetMathLibrary::Multiply_FloatFloat(bpfv__CallFunc_Sqrt_ReturnValu
e2__pf, bpfv__CallFunc_Sqrt_ReturnValue1__pf);
            bpfv__CallFunc_Divide_FloatFloat_ReturnValue__pf        =
FCustomThunkTemplates::Divide_FloatFloat(bpfv__CallFunc_Add_FloatFloat
_ReturnValue7__pf,
bpfv__CallFunc_Multiply_FloatFloat_ReturnValue6__pf);
            bpv__Cos__pf                                            =
bpfv__CallFunc_Divide_FloatFloat_ReturnValue__pf;
        }
   case 13:
        {
            bpfv__CallFunc_Square_ReturnValue__pf                   =
UKismetMathLibrary::Square(bpv__Cos__pf);
            bpfv__CallFunc_Subtract_FloatFloat_ReturnValue__pf      =
UKismetMathLibrary::Subtract_FloatFloat(1.000000,
bpfv__CallFunc_Square_ReturnValue__pf);
            bpfv__CallFunc_Sqrt_ReturnValue__pf                     =
UKismetMathLibrary::Sqrt(bpfv__CallFunc_Subtract_FloatFloat_ReturnValu
e__pf);
            bpv__Sin__pf = bpfv__CallFunc_Sqrt_ReturnValue__pf;
        }
   case 14:
        {
            bpv__LineTraceArcTimerHandle__pf                        =
bpfv__CallFunc_K2_SetTimer_ReturnValue__pf;
            __CurrentState = -1;
            break;
        }
   case 15:
        {
            bpv__znak__pf = 1.000000;
```

```
                __CurrentState = 12;
                break;
            }
        default:
            break;
        }
    } while( __CurrentState != -1 );
}
void      Aturret_C__pf1690381860::bpf__ExecuteUbergraph_turret__pf_1(int32
    bpp__EntryPoint__pf)
{
    float bpfv__CallFunc_Multiply_FloatFloat_ReturnValue1__pf{};
    float bpfv__CallFunc_Add_FloatFloat_ReturnValue__pf{};
    float bpfv__CallFunc_FClamp_ReturnValue__pf{};
    FRotator
    bpfv__CallFunc_MakeRotator_ReturnValue__pf(EForceInit::ForceInit);
    check(bpp__EntryPoint__pf == 22);
    bpv__AxisVerticalValue__pf = b01__K2Node_InputAxisEvent_AxisValue1__pf;
    FRotator  __Local__13 = FRotator(0.000000,0.000000,0.000000);
    bpv__VerticalRotationValue__pf    =    ((::IsValid(bpv__gun__pf))    ?
    (bpv__gun__pf->RelativeRotation) : (__Local__13));
    // optimized KCST_UnconditionalGoto
    UKismetMathLibrary::BreakRotator(bpv__VerticalRotationValue__pf,
    /*out*/          b01__CallFunc_BreakRotator_Roll2__pf,          /*out*/
    b01__CallFunc_BreakRotator_Pitch2__pf,                          /*out*/
    b01__CallFunc_BreakRotator_Yaw2__pf);
    bpfv__CallFunc_Multiply_FloatFloat_ReturnValue1__pf                  =
    UKismetMathLibrary::Multiply_FloatFloat(bpv__AxisVerticalValue__pf,
    0.100000);
    bpfv__CallFunc_Add_FloatFloat_ReturnValue__pf                        =
    UKismetMathLibrary::Add_FloatFloat(b01__CallFunc_BreakRotator_Pitch2__
    pf, bpfv__CallFunc_Multiply_FloatFloat_ReturnValue1__pf);
    bpfv__CallFunc_FClamp_ReturnValue__pf                                =
    UKismetMathLibrary::FClamp(bpfv__CallFunc_Add_FloatFloat_ReturnValue__
    pf, bpv__MinVertical__pf, bpv__MaxVertical__pf);
    bpv__RotationGunValue__pf = bpfv__CallFunc_FClamp_ReturnValue__pf;
    // optimized KCST_UnconditionalGoto
```

```
    UKismetMathLibrary::BreakRotator(bpv__VerticalRotationValue__pf,
    /*out*/          b01__CallFunc_BreakRotator_Roll2__pf,          /*out*/
    b01__CallFunc_BreakRotator_Pitch2__pf,                        /*out*/
    b01__CallFunc_BreakRotator_Yaw2__pf);
    bpfv__CallFunc_Multiply_FloatFloat_ReturnValue1__pf                  =
    UKismetMathLibrary::Multiply_FloatFloat(bpv__AxisVerticalValue__pf,
    0.100000);
    bpfv__CallFunc_Add_FloatFloat_ReturnValue__pf                        =
    UKismetMathLibrary::Add_FloatFloat(b01__CallFunc_BreakRotator_Pitch2__
    pf, bpfv__CallFunc_Multiply_FloatFloat_ReturnValue1__pf);
    bpfv__CallFunc_FClamp_ReturnValue__pf                                =
    UKismetMathLibrary::FClamp(bpfv__CallFunc_Add_FloatFloat_ReturnValue__
    pf, bpv__MinVertical__pf, bpv__MaxVertical__pf);
    bpfv__CallFunc_MakeRotator_ReturnValue__pf                           =
    UKismetMathLibrary::MakeRotator(b01__CallFunc_BreakRotator_Roll2__pf,
    bpfv__CallFunc_FClamp_ReturnValue__pf,
    b01__CallFunc_BreakRotator_Yaw2__pf);
    if(::IsValid(bpv__gun__pf))
    {
      bpv__gun__pf-
    >USceneComponent::K2_SetRelativeRotation(bpfv__CallFunc_MakeRotator_Re
    turnValue__pf,                    false,                     /*out*/
    b01__CallFunc_K2_SetRelativeRotation_SweepHitResult__pf, false);
    }
    return; // KCST_GotoReturn
}
```