Grabber Component:

Grabber:

Create a default pawn BP

(play, eject, select, blueprint menu, convert selected actor to blueprint)

Create Game Mode BP based on c++ class

(set BP Game Mode to default, with pawn BP)

Create C++ class Actor Component - grabber

Add Component to BP\_Pawn, custom class, add grabber

// Pseudo code back in Grabber .cpp //

// In Tick - get player view point //

declare FVector, FRotator,

pass in these to PlayerController::GetPlayerViewPoint()

#define OUT right below #includes

// DrawDebug - simulate Ray Cast //

Declare float Reach, FVector LTStart, LTDirection, LTEnd

LineTraceStart = PlayerViewLocation

LineTraceDirection = PlayviewRotation.vector();

LineTraceEnd = LTStart + (LTDirection \* Reach);

DrawDebugLine(GetWorld(), LTStart, LTEnd, FColor::Red, false, 0.0f, 0, 10.0f)

// last 4 params - persist line bool, time persist f, depth priority int, thickness f

\* Set Simulate physics on objects you want to be grab, set collision to physics, and overlap true

// LineTracebychannel will work but will ignore physics body filter

// FCollisionObjectQueryParams - Object Types Query interested (Object Trace Channel)

// FCollisionQueryParams - TraceParams(FName(TEXT("")), false, GetOwner())

// TraceParams (traceTag not hitTag, collisiontype(complex or simple), actor ignore)

// Use 5th override for ObjectQueryParams and 2nd override for QueryParams

// FCOQP vs FCQP – FCOQP looks for object trace channel (object type based on collision preset) for which the trace will hit and return actor that belongs to that object type and

Query Params is a struct for holding parameters defining collision functions (simple vs complex)

Declare FHitResult HitResult, FCollisionQueryParam TraceParams

GetWorld()->LineTraceSingbleByObjectType(OUT Hit Result, LTStart, LTEnd,

FCollisionObjectQueryParam(ECollisionChannel::ECC\_PhysicsBody), QueryParams)

AActor\* ActorHIt = Hit.GetActor();

Create a function in grabber that takes this Trace out of Tick, and then calls it.

Physics Handle part of Grabber Class:

Find Components by Class Grabber: <Need Physics Handle>

Add Component to pawn, Physics Handle

In grabber.h -

UPhysicsHandleComponent\* PhysicsHandle = nullptr;

UInputComponent\* InputComponent = nullptr;

// In BeginPlay look for attached physics handle

PhysicsHandle = GetOwner()->FindComponentByClass<UPhysicsHandleComp>();

InputComponent = GetOwner()->FindComponentByClass<UInputComponent>();

if (!PhysicsHandle) {return;}

if (!InputComponent){return;}

// Go to Input Binding, Project Settings, Action Map "Grab" to right click

InputComponent->BindAction("Grab, IE\_Pressed, this, <point to function>)

InputComponent->BindAction("Grab, IE\_Released, this, <point to function>)

// Create Grab and Release functions - void Grab(); void Release(); //

// Release () {PhysicsHandle->ReleaseComponent();}

void Grab ()

// try to grab actors on physics body object type, attach physics handle

// Declare 3 vars: Hitresult, componentHit, actorhit

hit result =Call GetActorHit Function() - raycast function that returns hit

auto ComponentToGrab = HitResult.GetComponent();

auto ActorHit = HitResult.GetActor();

If (ActorHit != nullptr)

{

PhysicsHandle->GrabComponent (ComponentToGrab, NAME\_None,

ComponentToGrab->GetOwner()->GetActorLocation(), true)

(

// Tick // if physics handle is attached, move each object holding, else nothing

if(PhysicsHandle->GrabbedComponent)

PhysicsHandle->SetTargetLocation(LineTraceEnd) // Need to reuse code