

Pseudocode

Guard:

OnPawnSee (*pawn)

{

if we already have a target only switch if
this new one is closer

if (current target dist < new target dist)

return;

else

target = pawn;

}

Determine Guard State()

{ if we have a current target

if (pawn sensing component has line of sight to target)

set guard state to attack

else

set guard to ~~patrol~~ search

else

set guard to patrol

}

Set Guard State (new state)

{ if (new state == current state)

return;

set the state

}

Guard

HandleAI()

3

Patrolling:

```
if we don't have a waypoint  
FindNewWaypoint();
```

if we have a waypoint and it is in the same room as us
get the distance to the waypoint

if we are close enough
FindNewWayPoint()

else

FindNewWaypoint();

Searching:

go to the player's last known position

if we can't find them check their last used door

If we still can't find them give up after a timer

Attack:

Walk straight at the target

3

Guard

is PatrolPoint InRoom ()

{
if we're actually in a room & point != nullptr
check

grab the door array from this room

if one of these doors is our point

return true

else

return false

}

FindNewPatrolPoint ()

{

if we're in a room (get our room)

get the room's array of doors

randomly select one

set our waypoint to it.

}

BaseCharacter

PerformRaycast (TraceProfile, FHHitresult)

{

perform a raycast from the character's viewpoint
against the TraceProfile

return the boolean result

}

Interact ()

{

if we're already interacting

drop our held object and set it to nullptr

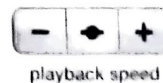
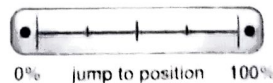
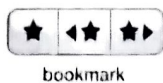
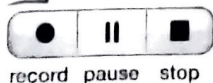
we're no longer interacting

else

if Perform raycast succeeds

grab what we hit

}



Base Character

Grab(AHitresult)

{

if the actor has the "grabbable" tag
cast it to our grabbable actor
set `bisInteracting` to true
call pickup on the grabbable actor

}

Player Character

Interact()

{

Super.'Interact()

if super completed and we're still not interacting
PerformRayCast for loot
if it has the loot tag
tell our inventory to collect loot

}

PlaceTrap()

{

if we have any traps
if a raycast against blockables succeeds
tell our inventory to try placing a trap.

}

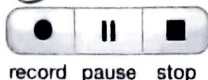
Base Trap Actor

HandleOverlap(Primitive, AActor, ...)

{

if we're not already triggered
if we're hitting what we're supposed to
cast as ABaseCharacter
if that works
set them as the target, Apply our Debuff, and `bisTriggered` is true

}



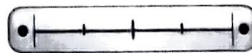
record pause stop



jump



bookmark



0% jump to position 100%



playback speed



volume

Slow/stop Trap Actor

ApplyDebuff()

{

store the target's original speed
alter their speed

start a timer to call RemoveDebuff()

}

RemoveDebuff()

{

set the target's speed back
Destroy self

}

AGrabbableStatic Mesh Actor

Tick()

{

if we are held && our character isn't null ptr

get the character's viewpoint (camera if they have one)

set our position to in front of ↑ but at the zoom distance

}

Pickup (ABaseCharacter) This is also called to drop

{

set our character to ↑

set bIsHeld to !bIsHeld

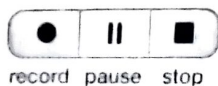
set bGravity to !bGravity

use these bools to set physics & gravity

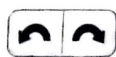
if bIsHeld = false

our character is null ptr now

}



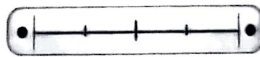
record pause stop



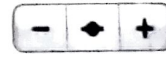
jump



bookmark



0% jump to position 100%



playback speed



volume

A Grabbable Static Mesh Actor

Throw()

set bisheld & gravity & collision

get our character's forward vector
add a large amount of force to our mesh in that vector
bWasThrown = true

OnHit(...)

if bWasThrown == true
Apply a lot of damage to the mesh to fracture it
start the despawn timer

Loot Actor

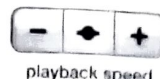
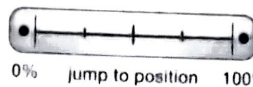
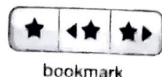
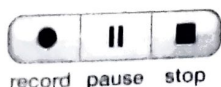
PostInitialize components()

clamp ~~the~~ Chance To Find Trap between 0 & 100

Die()

if Respawn Delay > 0
turn off the particles
hide the sphere in game
turn off collision
start a timer to call Respawn()
turn off tick

else
Destroy()



LootActor
Respawn()

{
reverse what Die() did
}

InventoryComponent

DidFindTrap(Lootactor * loot)

{
if loot > chance to find trap is ~~100~~ >= a random number 1-100
return true
return false
}

AddRandomTrap()

{
for each inventory slot
if this slot is empty
choose a random trap
increment trap count
set this slot to the random trap
break
}

CollectLoot(AActor * loot)

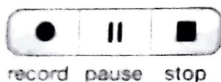
{
if DidFindTrap(loot) & HasOpenSlot()
AddRandomTrap()

loot -> Die()

return a random score value for the loot

return 0; (if it wasn't loot)

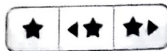
}



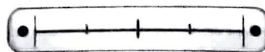
record pause stop



jump



bookmark



0% jump to position 100%



playback speed



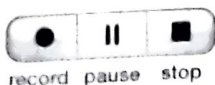
volume

Inventory Component
PlaceTrap(FVector location)

if this inv slot isn't empty & trapcount > 0

spawn a trap of the slot's type at the location
set the trap's owner to our owner
set this inv slot to empty
decrement trapcount

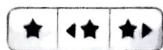
3



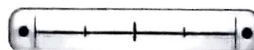
record pause stop



jump



bookmark



0% jump to position 100%



playback speed



volume