Pseudocode

Guard:

```
OnPawnSee (*pawn)
{
  if we already have a target only switch if
  this new one is closer
  if (currenttargetdist < new target dist)
      return;
  else
      target = pawn;
}

DetermineGuardState()
{ if we have a current target
    if (Pawnsensingcomponent has line of sight to target)
       setguardstate to attack
    else
       set guard to ~~patrol~~ search
  else
     set guard to patrol
}

Set GuardState (newstate)
{ if (newstate == currentstate)
     return;

  set the state
}
```

```
Guard
HandleAI()
{
 Patrolling:

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

    if we have awaypoint 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

}
```

Guard

```
isPatrolPointInRoom()
{
  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 (getour room)
   get the room's array of doors
    randomly select one
     set our waypoint to it.
}
```

BaseCharacter

```
PerformRaycast(TraceProfile, FHitresult)
{
  perform a raycast from the character's wiewpoint
  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
}
```

BaseCharacter

```
Grab(FHitresult)
{
    if the actor has the "grabbable" tag
        cast it to our grabbable actor
        set bIsInteracting to true
        call pickup on the grabbable actor
}
```

PlayerCharacter

```
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.
}
```

BaseTrapActor

```
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
}
```

```
Slow/stop TrapActor
ApplyDebuff()
{
    store the target's original speed
    alter their speed
    start a timer to call RemoveDobuff()
}

RemoveDebuff()
{
    set the target's speed back
    Destroy self
}
```

AGrabbableStaticMeshActor

```
Tick()
{
    if we ar held && our character isn't nullptr
      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 t !bGravity
    use these bools to set physics & gravity

    if bIsHeld = false
       our character is nullptr now

}
```

AGrabbableStaticMeshActor

```
Throw()
{
    set bisheld & gravity & collision

    get our character's forwardvector
    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
}
```

LootActor

```
PostInitialize components()
{
    clamp ~~the~~ ChanceToFindTrap between 0 & 100
}

Die()
{
    if RespawnDelay > 0
        turn of the particles
        Hide the sphere ingame
        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->chancetofind trap is ~~[illegible]~~ >= a random number 1-100
     return true
  return false
}
AddRandomTrap ()
{
   For each inventory slot
      if this slot is empty
        choose a random trap
        increment trapcount
        ~~Al~~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
}
```