Dayton – Pseudocode Version 2

**ALootingLootersGameModeBase**

ALootingLootersGameModeBase()

{

LOAD all room blueprints

LOAD all asset blueprints

LOAD all loot blueprints

LOAD all door blueprints

}

ARoomActorBase\* GetRoomActorIsIn(AActor\* actor)

{

GET all rooms from GameState

FOR EACH room

Find if actor is in room

IF true

RETURN that room

ELSE

Continue

ENDIF

}

TSubclassOf<AActor> GetRandomAssetOfTypes(TArray<FString> types)

{

FOR EACH Asset in Asset array

IF Asset has same types as type parameter array

ADD to list of viable meshes

ENDIF

RETURN a random mesh from the list of viable meshes

}

TSubclassOf<AActor> GetARandomLootAsset()

{

RETURN a random Loot blueprint from list of Loot blueprints

{

Void RespawnPlayer(APlayerController\* controller, uint8 team, FTransform location)

{

GET a PlayerStart

SPAWN a default pawn for controller

IF pawn spawned successfully

SET pawn team

SET pawn location to transform

MULTICAST set mesh material on pawn

MULTICAST apply material to mesh on pawn

ENDIF

}

Void Server\_StartEndgame()

{

GET all characters

FOR EACH character

MULTICAST disable character input

}

Void EndMatch()

{

GET all Player controllers

FOR EACH controller

CALL controller to client travel to main menu

}

Void HandleNewPlayer(APlayerController\* controller)

{

GET pawn from controller

ASSIGN pawn team

ASSIGN pawn material

}

**ALootingLootersGameStateBase**

Virtual void BeginPlay() override

{

SERVER CALL Generate Rooms

SERVER CALL Populate Rooms with assets

SERVER CALL Connect Room doors

SERVER CALL Generate loot in rooms

SET Match timer

}

Virtual void Tick (float deltatime) override

{

REDUCE match timer by deltatime

}

Void Server\_GenerateRandomRoomLayout()

{

GET all room blueprints

GET int “total rooms to generate”

FOR LOOP to make each room

GET a random room from the blueprints

CALCULATE spawn offset based on index value

SPAWN Room

ENDFOR

}

Void Server\_PopulateRoomSockets()

{

FOR EACH room in Room array

CALL PopulateSockets on room

}

Void Server\_GenerateRandomRoomConnections()

{

FOR EACH room in Room array

CALL GenerateDoorConnections on room

}

Void Server\_GenerateLoot()

{

FOR EACH room in Room array

CALL GenerateRandomLoot on room

}

Void Server\_EndGame()

{

GET GameMode

CALL EndMatch on GameMode

}

Void Server\_StartEndGame()

{

GET GameMode

CALL StartEndGame on GameMode

SET buffer Timer to call EndGame

}

**ARoomActorBase**

Void SetRoomMesh(AStaticMeshActor\* mesh)

{

IF AUTHORITY

SET mesh to parameter mesh

SET box collision to fit mesh

ENDIF

}

Void PopulateEmptySockets()

{

IF AUTHORITY

GET all meshes in room

FOR EACH mesh in mesh array

FOR EACH socket on mesh

GET socket name

CONVERT socket name into types

IF TYPE is door

CALL Server\_SpawnDoor

ELSE

CALL Server\_SpawnAsset

ENDIF

ENDFOR

ENDIF

Void Server\_GenerateDoorConnections()

{

GET GameMode

GET all rooms from GameMode

GET all doors already connected

FOR EACH connected door

IF connection is a room inside the room array

CONTINUE

ELSE if room is our room

CONTINUE

IF room is unconnected

ADD to list of unconnected rooms

ENDFOR

GET all unconnected doors for this room

FOR EACH unconnected door

GET a random door

CONNECT door to random room door

REMOVE random room from list of unconnected rooms

ENDFOR

}

Void Server\_SpawnDoor(FVector SpawnLocation, FRotator SpawnRotation)

{

GET GameMode

GET Door blueprint from GameMode

SPAWN Door at SpawnLocation

ADD door to list of doors

}

Void Server\_SpawnAsset(TArray<FString> Types, FVector SpawnLocation, FRotator SpawnRotation)

{

GET GameMode

GET random asset blueprint from GameMode (GetRandomAssetOfTypes)

SPAWN the asset

IF asset is grabbable

ADD to array of grabbable assets

ELSE

ADD to array of static assets

ENDIF

}

**ADoorActor**

Virtual void Tick(float deltatime) override

{

IF connected to another door

SET mesh visible

SET collision active

ELSE

SET mesh invisible

SET collision inactive

DISABLE ticking

ENDIF

}

Void ApplyConnection(ADoorActor\* otherdoor)

{

SET connection to otherdoor

ENABLE ticking

}

Void TeleportPawnToOtherDoor(ONOVERLAP DELEGATE)

{

IF connected to another door

IF collided actor is a character

SET player location to other door location

ROTATE player to other door direction

IF character is player controlled

ROTATE the controller

ENDIF

SET player door accessed to this

ENDIF

ENDIF

}

**AAssetTemplate**

Void PopulateLootSockets()

{

GET all meshes

GET GameMode

FOR EACH mesh in meshes

FOR EACH socket in mesh

IF socket contains “LOOT” type

GET loot blueprint from GameMode

SPAWN loot

ADD to list of loot

ENDIF

ENDFOR

ENDFOR

}

**ALootActor**

Void NetMulticast\_Die()

{

CALL Die

}

Void Die()

{

IF AUTHORITY

IF has respawn timer

SET hidden

DISABLE ticking

SET respawn timer

ELSE

DESTROY this

ENDIF

ENDIF

}

Void Respawn()

{

IF AUTHORITY

SET visible

ENABLE ticking

ENDIF

}

**ABaseTrapActor**

Void HandleOverlap(ONOVERLAP DELEGATE)

{

IF trap has not been triggered yet

IF other actor is not owner

SET trap team to owner team

IF other actor team is trap team

RETURN

ELSE

IF mesh activates

SWAP mesh to activated mesh

ENDIF

SET target to other actor

APPLY debuff to other actor

ENDIF

ENDIF

ENDIF

}

Void Server\_PlaySound()

{

MULTICAST play trap sound

}

Void NetMulticast\_PlaySound()

{

SPAWN sound at trap location

}

Void ApplyDebuff()

{

IF AUTHORITY

SERVER play sound

ENDIF

APPLY debuff effect to player (DEPEND ON TRAP TYPE)

Set RemoveDebuff timer

}

Void RemoveDebuff()

{

RESTORE lost stat to player

DESTROY this

}