**TANK GAME**

**Initial Setup**

-------------------------------------------------Initial Setup (Build/Push) --------------------------------------------

1. Setup GitHuB Repo
2. Create Landscape
3. Create Landscape layered material
4. Paint Landscape material
5. Add sockets to mesh

• Create Socket

• Select mesh in preview

• move mesh into position

• clear mesh from preview

• rename socket

1. Create Blueprint BP\_TankPawn based on a pawn blueprint
   * Add Rotation controls spring arm (disable spring arm inherit roll)
   * Add Look controls scene comp.
2. Create HUD BP and UI Crosshair point
3. Create Main Menu Level / UMG Widget
4. Create C++ Tank.h class, re-parent to BP\_TankPawn
5. Create C++ PlayerController class
   * #include Tank.h
   * GetControlledTank() const { return Cast<ATank>(GetPawn());}
6. Create BP based on C++ PC class
7. Create GameMode BP based on C++ class
8. Create an AI Controller C++ class (other classes) #include tank.h, GetAITank(), GetPlayerTank()
9. Configure BP\_TankPawn, Switch AI Controller Class to the new AI Controller C++ class
10. Use Debug Logs to verify that Player knows itself, AI knows itself, and AI knows the player

------------------------------------------------------- (Build/Push) -----------------------------------------------------------

**AimTowardCrossHair function that initializes aiming in PlayerController**

1. In TankPlayerController, create the methods for aiming – **void AimTowardsCrosshair()**{}

// Check if GetControlled tank is valid

// GetWorld Location of line trace through the crosshair

// if it hits the landscape, tell Controlled tank to aim at this point

1. Create a sub-method of AimTowardsCrosshair, GetRayHit() location using a FVector & out @@parameter and returns a bool
2. Create a sub-method of GetRayHit(), GetLookDirection using out @@parameter to mutate look direction
3. De-project the position of the dot on the screen to world coordinates (overview next steps)
   1. Find Crosshair position.
   2. De-project screen world coordinate (screen position of the cursor to world direction).
   3. Line trace along that direction (look direction).

**Void AIMTowardsCrosshair()** is Parent Method responsible for shooting/aiming, called in Tick.

Declares an FVector that will be used as a mutator (aka OUT\_@@param)

Checks if the *GetRayHit()* function that passes in that out Vector is true.

if true, then the Tank.cpp AimAt function is called passing in that out Vector.

**Bool GetRayHit (FVector &Location\_OUT) –** Creates ScreenLocation and calls *LookDirection()*

The first thing it does is get the Screen Location cords

* + Create 2 UPROPERTY float members associated with player ui’s Anchor’s percentage location values to where the dot image is located in the widget.
  + Declares 2 int out @@parameters, passes those into GetViewportSize.
  + Declare ScreenLoc FVector2D that multiplies the out int @param by the member floats

Create another sub-function that also uses an out @param vector for getting Look direction. Above *GetLookDirection()*, create an FVector that will be used as an out @param. Check *GetLookDirection()* is true, and then call *GetVectorHitLocation()*

**Bool GetLookDirection(ScreenLocation, FVector &LookDirection)** - uses DeprojectScreenLocation method to mutate Look Direction

Declare an FVector LookLocation

Return DeprojectScreenLocation(Screen.X, screen.Y, LookLocation, LookDirection) – true or false

**Bool GetVectorHitLocation(FVector LookDirection, FVector &HitLocation\_OUT)** –

Create a UProperty float for LineTraceLength. Create 2 FVectors for start and end, and an FHitResult var

To get Line trace start, you need to call PlayerCameraManager->*GetCameraLocation()*;

Call GetWorld()->LineTraceSingleByChannel(FHitResult, tracestart, traceend, ECC\_Visibility);

Check for actor. If Actor is not nullptr, then set HitLocation\_OUT to the FHitResult.location.

**AIMING COMPONENT (AC) & AimAt function**

1. Create an Actor Component C++ class (TankAimingComponent) – declare one in tank.h + cpp
2. Create an void AimAt func in AimingComponent with FVector Hit and float launchspeed @@params
3. Create an void AimAt func in tank.cpp that call the aiming components AimAt func
4. Create a tick function in AI controller, and have it track PlayerPosition using AimAt function
5. Add UE\_LOG function to Aiming Component to verify that its working correctly.
6. Set up properties and methods for AimingComponent:
   * Class UStaticMeshComponent\* Barrel = nullptr, Turret = nullptr;
   * Void SetBarrelReference(uStatic\*) and void SetTurretReference(ustatic\*)
   * Void MoveBarrel(FVector AimDirection)
7. Verify float launch speed is a @@param in AimingComponent’s AimAt(). Set launch speed in tank, but during call of AC’s AimAt() function, tank’s launch speed member variable is known directly.
8. Implement UGameplayStatics::SuggestProjectileVelocity in AC’s AimAt() function in order to get the recommended velocity vector and the unit vector (aka direction) of that velocity vector.
   * Declare 2 FVectors, OutLaunchVelocity and StartLocation before calling SPV().
   * If (UGameplayStatics::SuggestedProjectileVelocity(this, OutLaunchVelocity, StartLocation, Hit Location (@@param), LaunchSpeed (@@param), ESuggestProj::DoNotTrace)
   * Convert velocity vector into a normal vector (aka a unit vector to get the direction) -Auto aimDirection = OutLaunchVelocity.GetSafeNormal();
9. Create void MoveBarrel() function in AimComponent that passes in the the AimDirection whose purpose is to move the barrel the right amount this frame.
   * Get dot’s expected and barrel’s current unit vector’s direction converted into a rotation.
   * Calculate the difference as a deltaRotator = AimRot - BarrelRot
   * Give a max elevation speed, and the frame time (have barrel ref call elevate)
10. C++ class Barrel – create MaxDegreesPerSecond, MaxElevation, & MinElevation member vars.
11. Modify Barrel – add a meta to the class header to make it available to add in BP.
12. Repurpose Barrel in Aiming Component – replace UstaticMesh refs with UBarrel in C++ and BP.
13. Create an Elevate() in Barrel. Call function in AC’s MoveBarrel to change mesh location.
14. Create Turret class – Same thing as Barrel class except use rotation and yaw.

**Projectile**

1. Create AProjectile class
2. Create a Fire function in tank.cpp – SpawnActor<AProjectile>(blueprintprojectile, , , )
3. Add Projectile Movement