ToonTanks Technical Design

Programming by John McCoubrie

Overview

- Gameplay Overview
- Lessons Learned
- Class Setup
- Dodge Mechanic
- Shield Mechanic
- Tower Tracking
- Boss Tracking
- UI Overview
- Contact Info and Credits

Gameplay Overview

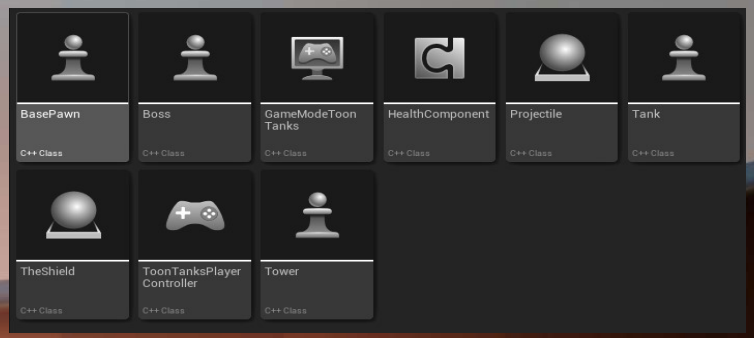
- ToonTanks is a tutorial-based game created with Unreal Engine 5 utilizing the “Unreal Engine 5 C++ Developer: Learn C++ and Make Games”
- Tutorial Features: Ability to move the tank and shoot a weapon with unlimited ammo at 2 static towers with basic AI functionality.
- Original Features: Dodge and shield mechanic, health system, UI system, 2 unique enemy types, updated AI tracking, game states, menu system.

Lessons Learned

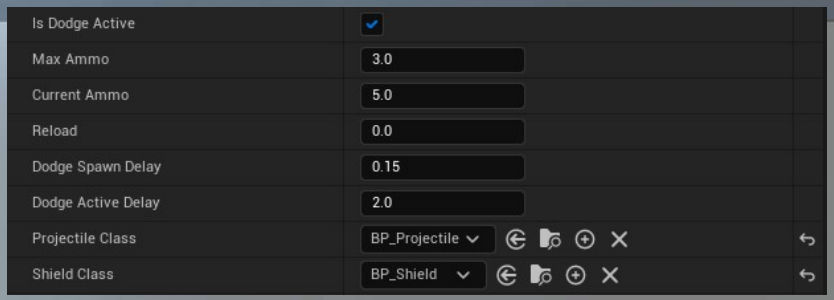
Issue	Solution
The boss would not properly track the player, he would move along the Z axis due to the VInterpConstantTo function.	Locked the Z Axis and updated collision information with the floor.
Game controller class contained all UI code.	Code became cluttered and hard to read/debug. Put the UI code in a UI manager class and created a class for HUD, enemy HUD, etc. to promote simplicity and reusability.
Unreal specific: don't force C++ code when unnecessary, sometimes blueprints is the answer.	Blueprints was more effective than scripting for UI elements and on-screen prompts.
Loaded an array with all enemies in the game in a Tick function.	The initial enemy array only needs to be in the begin play function (called once) and updated only when an enemy is destroyed, not every frame.
World location vs. Local location led to vast differences in expected output.	All pawns used world location. Remain consistent with which location types you will use, world location was easier to manage than local for each player/enemy type.

Class Setup

- Tank (player), Tower (enemy), Boss (enemy), all derive from BasePawn which adds basic functionality to each.
- BasePawn sets up all capsule components, base meshes, projectile spawn points, and shield spawn points.
- HandleDestruction handles all death animations, particles, sounds, etc. for all classes.
- Rotate turret and fire methods are created in BasePawn and customized in each player and enemy type.



Dodge Mechanic



- Player presses shift to dodge, 2 second respawn
- Dodge updates the speed setting for 2 seconds before returning the speed to normal as opposed to applying physics
  - Quicker to edit on the fly and less expensive programmatically
  - Compared to applying a force, felt more twitchy and fun to use
- Dodge delays can be changed in the editor
- Used bools to determine the dodge state for simplicity and timing


```
void ATank::StartDodge()
{
    //Timer calls end dodge after spawn dealy (.15 seconds) if the shield is active
    if (IsDodgeActive)
    {
        Speed = 2600;
        //IsDodgeActive = false;
        GetWorld()->GetTimerManager().SetTimer(DodgeTimerHandle, this, &ATank::EndDodge, DodgeSpawnDelay, false);
    }
}
//The dodge is ended and the respawn timer is set for 5 seconds before you can dodge again
void ATank::EndDodge()
{
    Speed = 800;
    if (IsDodgeActive)
    {
        IsDodgeActive = false;
        GetWorld()->GetTimerManager().SetTimer(DodgeRespawnHandle, this, &ATank::SetDodge, DodgeActiveDelay, false);
    }
}
```

Shield Mechanic

- Player presses right click to deploy a shield
- If the shield is hit, add +5 ammo, the shield is destroyed, and goes on a cooldown
- Used a delegate on the shield mesh to call SuccessfulBlock which uses methods AddAmmo() and UnBlock() when hit by a projectile
- Delegate used to separate functionality for ease of use


```
//Delegate that broadcasts the designated function whenever it is hit by the projectile
ShieldMesh->OnComponentHit.AddDynamic(this, &ATheShield::SuccessfulBlock);
```

```
//after a successful block, add dynamic calls successful block and ammo is added through the add ammo method on tank
void ATheShield::SuccessfulBlock(UPrimitiveComponent* HitComp, AActor* OtherActor, UPrimitiveComponent* OtherComp, FVector NormalImpulse, const FHitResult& Hit)
{
    UE_LOG(LogTemp, Display, TEXT("hit"));
    Tank->AddAmmo();
    Tank->UnBlock();
}
```

Tower Tracking

```
int32 AGameModeToonTanks::GetTargetTowerCount()
{
    TArray<AActor*> Towers;
    UGameplayStatics::GetAllActorsOfClass(this, ATower::StaticClass(), Towers);
    return Towers.Num();
}
```

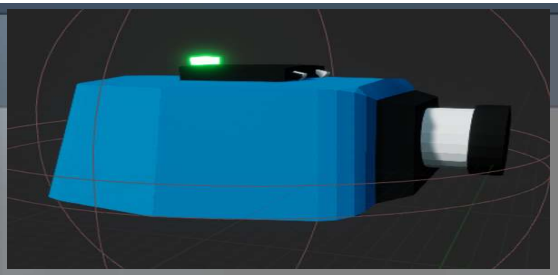
- GetTargetTowerCount() is set up in the game manager class and allows a running count of enemies in the game to be tracked
- If the player comes in range of a tower, rotate turret and Fire are called from Base Pawn
- Created a new method for check fire condition as opposed to adding code to BeginPlay()

```
void ATower::CheckFireCondition()
{
    if (Tank == nullptr)
    {
        return;
    }
    if (InFireRange() && Tank->bAlive)
    {
        Fire();
    }
}
```

```
bool ATower::InFireRange()
{
    if (Tank)
    {
        float Distance = FVector::Dist(GetActorLocation(), Tank->GetActorLocation());
        if (Distance <= FireRange)
        {
            return true;
        }
    }
    return false;
}
```

Boss Tracking

```
if (CurrentLocation.Y <= 500)
{
    TargetLocation.Y = 500;
    FVector NewLocation = FMath::VInterpConstantTo(CurrentLocation, TargetLocation, DeltaTime, BossSpeed);
    SetActorLocation(NewLocation);
}
else if (CurrentLocation.Y >= 500)
{
    TargetLocation.Y = -2000;
    FVector NewLocation = FMath::VInterpConstantTo(CurrentLocation, TargetLocation, DeltaTime, BossSpeed);
    SetActorLocation(NewLocation);
    UE_LOG(LogTemp, Warning, TEXT("Tracking"));
}
```



- Multiple issues attempting to get the Boss to move left to right (see commented code)
- Settled on moving the boss toward the player by using the VInterpConstantTo math function (below method)
- Had to freeze the Z axis as the boss is larger than the player which caused it to move down along the Z axis, getting stuck in the map.

```
FVector CurrentLocation = GetActorLocation();
FVector TargetLocation = Tank->GetActorLocation();
float BossSpeed = 350.f;
FVector NewLocation = FMath::VInterpConstantTo(CurrentLocation, TargetLocation, DeltaTime, BossSpeed);
SetActorLocation(NewLocation);
```

```
float BossSpeed = 350.f;
FVector CurrentLocation = GetActorLocation();

if (CurrentLocation.Y < -900)
{
    CurrentLocation += GetActorForwardVector() * BossSpeed * DeltaTime;
    SetActorLocation(CurrentLocation);
}
```

UI Overview

- UI created using Widget blueprints
- Widget blueprints talk to the Tank, Tower, and Boss classes to access value that update the UI
- Start, Pause, and Game Over menu are all handled through the GameController class
- Enemies have health bars above their head that move with them and dynamically update

Contact Info and Credits

- Github source code
  - https://github.com/john-mccoubrie/ToonTanks
- Portfolio Link
  - Check out my other projects and gameplay demos here
  - https://john-mccoubrie.github.io/
- Itch.IO
  - Playable game
  - https://johnmccoubrie.itch.io/
- Reach out
  - LinkedIn: https://www.linkedin.com/in/john-mccoubrie/
  - Email: Jmccoubrie4@gmail.com
  - Phone: (610) 908 – 6115
- Tutorial used as base game
  - https://www.udemy.com/course/unrealcourse/learn/lecture/31734150#overview