Section 4 Slides - Battle Tank

<< Back To Section 3

To Section 5 >>

These are the slides that accompany the Complete Unreal Developer Course.

See me develop the slides as I write the course…

- Right click or Insert > Comment to comment, especially if you see a typo
- The slides will update immediately as I change things.

Enjoy your stay!

Ben Tristem

Intro, Notes & Assets

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Battle Tank is an open world tank fight
- This will be a head to head battle
- Other players can be human or simple AI
- Heavy focus on control systems
- Also learning terrains, UI, terrain sculpting & more
- Dive right in and enjoy yourself!

Research Tank Games

- Watch at least 1 YouTube video of a tank moving
- Watch at least 1 YouTube video of a tank game
- Play World of Tanks (free, multi-platform)
- Take notes on controls, features & challenges
- Share your notes with us.

Battle Tank Overview

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- An overview of the finished Battle Tank project.

Game Design Document (GDD)

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- The Concept, Rules and (initial) requirements
- We'll iterate around a loop while making this game
- Constantly asking "what's least fun"
- Remember we're not AAA studios
- Let's find the essence of fun of this game.

Concept

Tank battle is an open-world head-to-head tank combat game.

The terrain will be used for tactical advantage.

The focus will be on flow and feel.

Rules

- You can move anywhere in the terrain, which is surrounded by mountains
- Both players start with finite health and ammo
- Each direct hit takes away health
- The last player standing wins.

Requirements

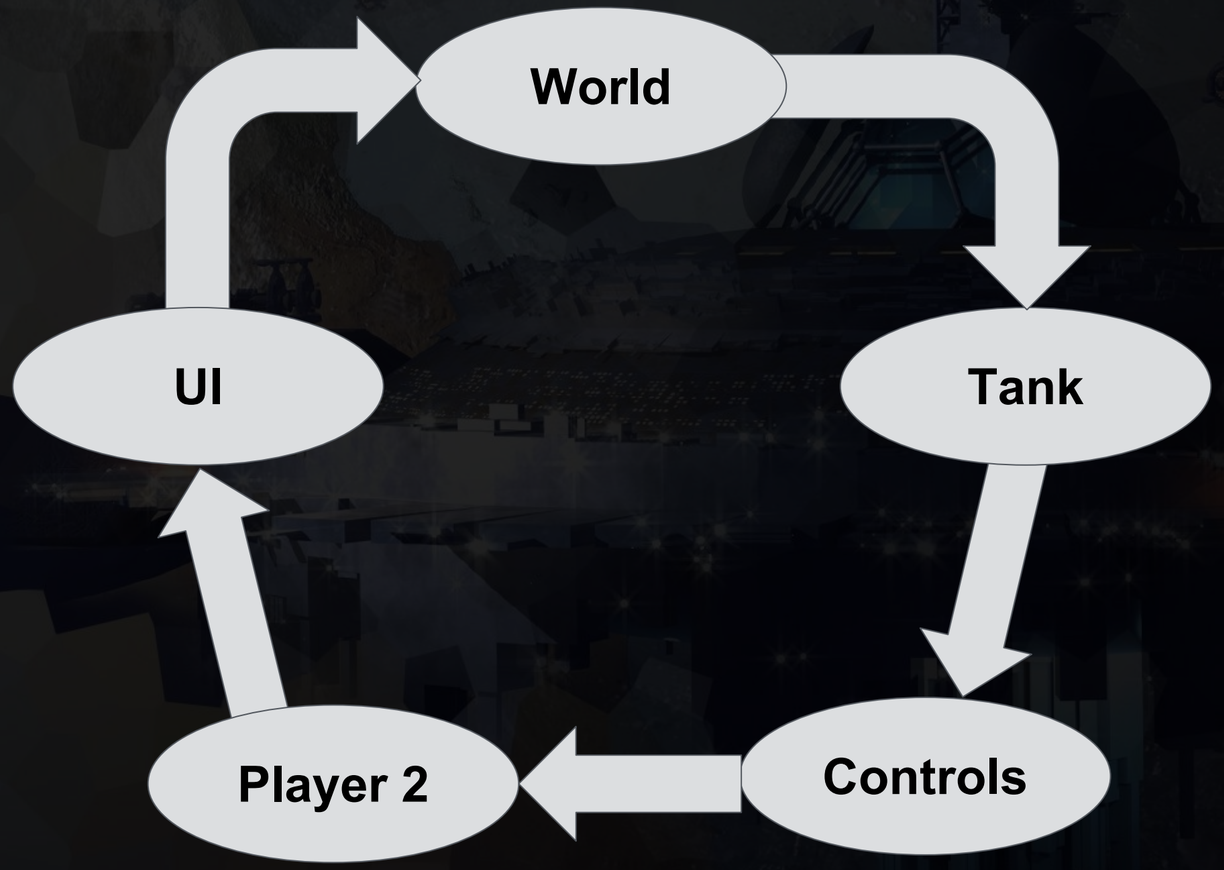
SFX: Gun firing, explosion, barrel moving, turret moving, engine sound.

Static mesh: Simple tank comprising tracks, body, turret and barrel.

Textures: Later-on we'll want to add for visual flare.

Music: Background music to create tension.

Our Iterative Cycle



Write & Share Your GDD

- Write your understanding of the **Concept**
- Write out the **Rules** you can think of
- List the asset **Requirements** (graphic, sound, etc)
- Share your work with us.

Setting Up a GitHub “Repo”

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Creating an online repository for your project
- GitHub provides public hosting for free
- We will use their default UnrealEngine .gitignore
- We'll then "clone" this repository to our machine
- How to use a readme.md with markdown*

https://daringfireball.net/projects/markdown/basics

Setup Your Remote Repo

- Setup a repo on GitHub (or BitBucket)
- Write your first readme.md
- Why not include your GDD in there?
- Share the repo link with us.

Creating & Deleting Landscapes

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

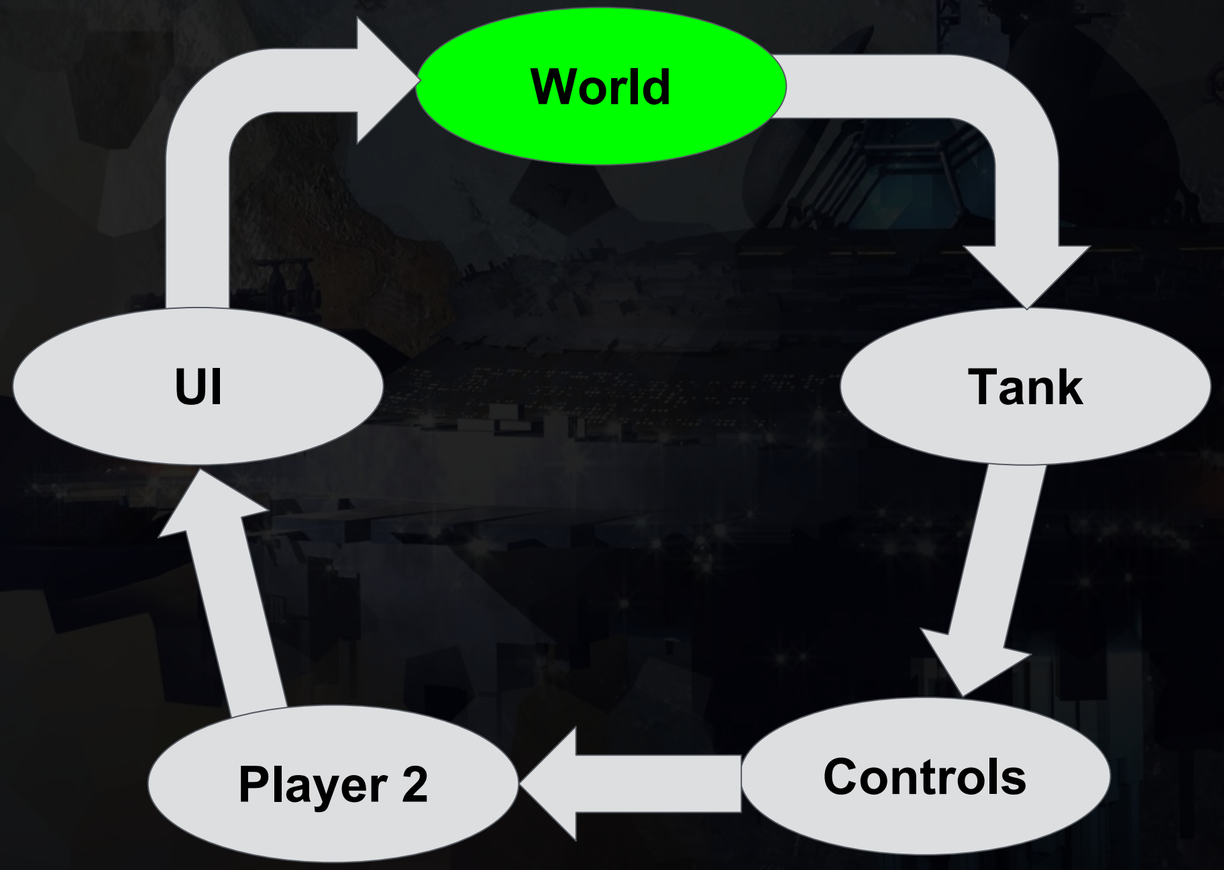
- Creating an Unreal project in an existing “repo”
- What’s good about Landscapes in Unreal Engine
- How to add a Landscape in Unreal
- How to delete a Landscape in Unreal.

What’s Good About Landscapes

- ⅙ or less of storage of Static Mesh
- Supports Level Of Detail (LOD) & streaming
- Up to 8196x8196 heightmaps for import*
- Good built-in tools for sculpting
- Work well with mini-maps

** For example Terragen or World Machine*

Our Iterative Cycle



Try and Add a Landscape

- See if you can find where to add it
- Start without using a web search or docs
- Use any settings, we'll delete and re-add
- Share how long it took you to find.

Landscape Setup & Scaling

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- You can change position & rotation later
- Scale will impact terrain size, so set on creation
- How to choose your “Section Size”
- The effect of the “Number of Components”
- Creating a landscape of a specific scale.

Create Your Landscape

- Make it about 1000m square
- Make each quad 0.5m in size
- Use 8x8=64 components (like a chess board)

A Landscaping Process

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

1. **Sculpt:** hills, valleys & flat areas
2. **Smooth, flatten & ramp:** create useful features
3. **Erosion & noise:** make it more organic
4. **Paint:** use layered materials
5. **Details:** add details (foliage, trees, etc)

Create your Version 1 Landscape

- Remember you'll iterate again later
- Keep it simple but believable looking
- Have a 500x500m flat area in the middle for now
- This will allow simple movement & raycast testing.

Upgrading Engine Version

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Epic games launcher helps manage versions
- Remember to commit your project first
- You can then “Convert in-place”
- Check your project runs OK in new version
- Close everything and re-commit
- How to tag a commit in GitHub.

What’s New in Unreal 4.11

Relevant to us at the moment…

- Faster lighting build times
- Runtime performance enhancements

For a full list of improvements...

https://www.unrealengine.com/blog/unreal-engine-4-11-released

Upgrade Your Project

- Go through the upgrade process
- Ensure the new project runs ok
- Commit the changes
- Optionally: tag the specific commit with “v.4.11”

Using Landscape Layers

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

Paint Your Landscape

- Create a material for your landscape
- Set **Usage > Used with Landscape**
- **LandscapeLayerBlend** node & Vector Parameters
- Add at at least two layers & create LayerInfo
- Paint the landscape from the Modes tab
- Screenshot and share with us.

Flat Shading Low Poly Landscapes

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Unreal’s tools are setup for photoreal landscapes
- Once you set the bar high, the rest must match
- An alternative is to opt for a low-poly look...
- ...then you can focus on gameplay, story, sound
- Can be a good choice for smaller teams
- How to make low-poly, flat-shaded landscapes.

Choose Your Final Landscape Style

- Decide which route you want to pursue
- If high-poly, use this time to tweak
- If low-poly, then create your initial landscape
- Remember to leave a flat area in the middle.

More Landscaping Tools

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- How to make flat shading optional
- Importing and exporting landscape heightmaps
- Reducing the resolution of a landscape
- Using a texture in a landscape material.

Try Adding a Texture

- Add a texture to your high (or low) poly landscape
- Paint & share

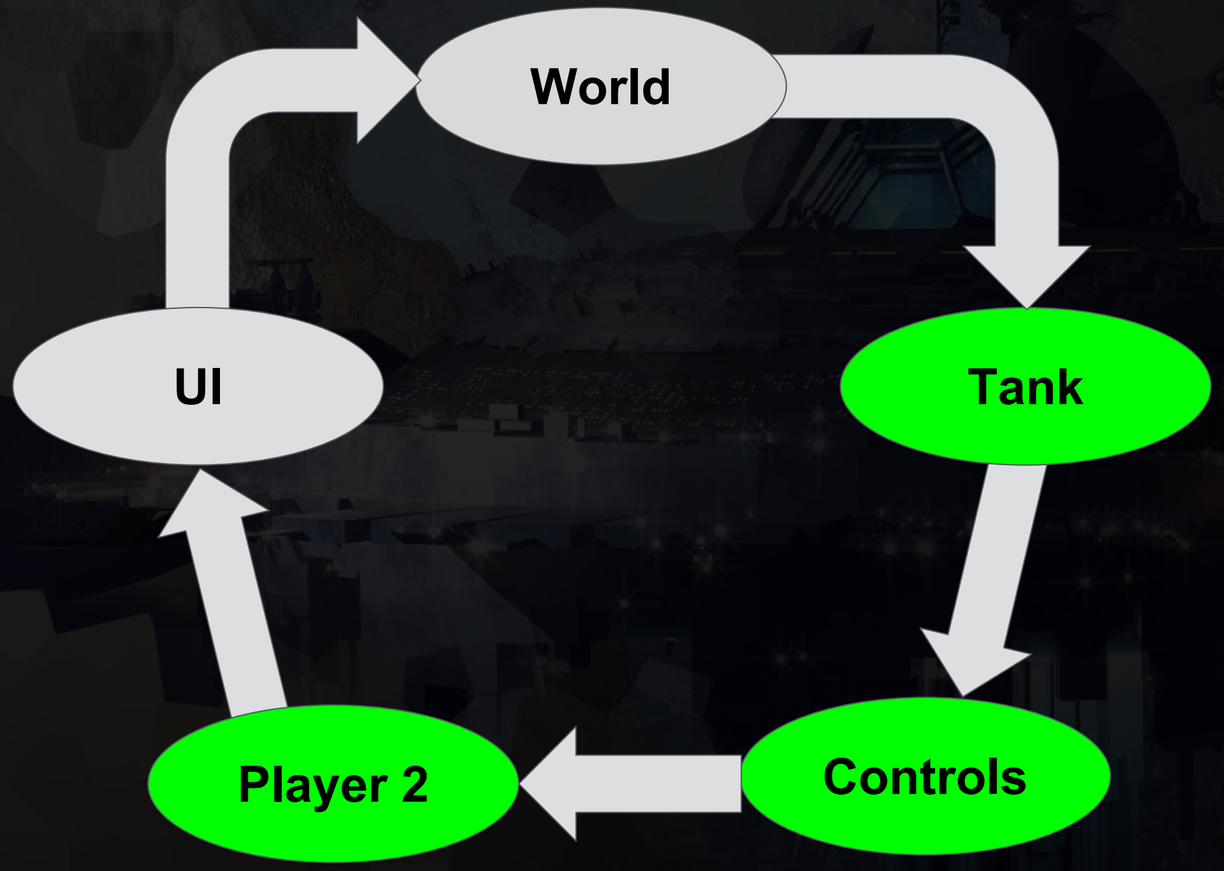
Tank Control System

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

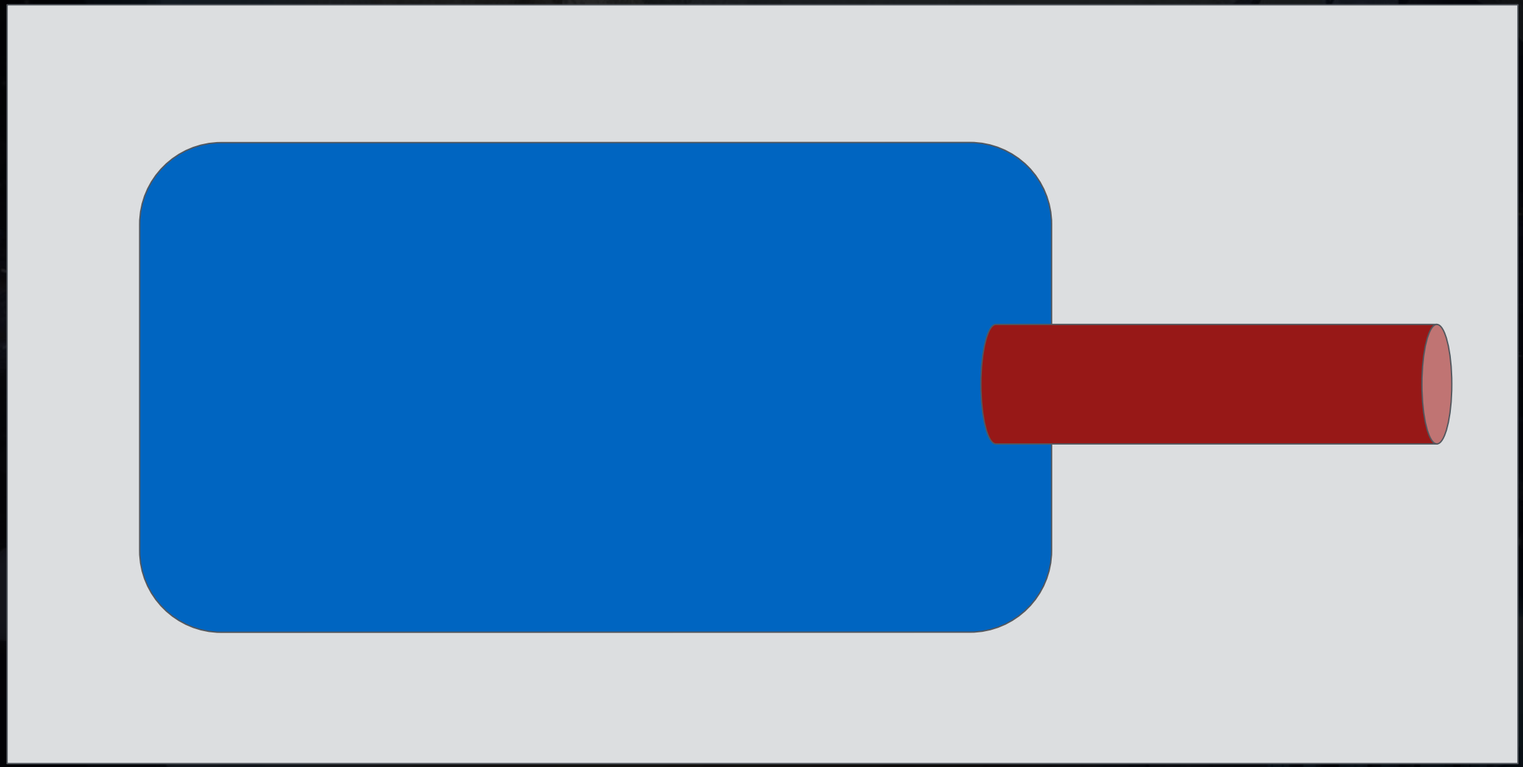
- Support keyboard, mouse & gamepad controller
- Mapping player intentions to control inputs
- Mapping control inputs to actor actuators
- Introducing the concept of “fly by wire”.

Our Iterative Cycle



Control Inputs

Major Tank Components



Virtual Control & “Fly-by-wire”

Intention	Manual Control	Fly-by-Wire Control	Actuator(s)
Forward	Symmetric Triggers	Left Stick Forward or W Key	Both Tracks Forward - Same Speed
Backwards / Braking	Symmetric Bumpers	Left Stick Backward or S Key	Both Tracks Backwards - Same Speed

Complete the Control Table

- Finish this control intention table
- There is no right or wrong here
- The idea is to think through the consequences
- Fly-by-wire controls only on KB / mouse
- Manual control only with controller.

Virtual Control & “Fly-by-wire”

Intention	Manual Control	Fly-by-Wire Control	Actuator(s)
Forward	Symmetric Triggers	Left Stick Forward or W Key	Both Tracks Forward - Same Speed
Backwards / Braking	Symmetric Bumpers	Left Stick Backward or S Key	Both Tracks Backwards - Same Speed
Strafe	N/A	N/A	N/A “You Can’t Strafe a Tank Fool”
Rotate Body	A-Sym Triggers / Bumpers	Left Stick Left / Right or A/D Keys	Tracks - Different Speeds
Rotate Turret	N/A	Look & Aim with Right Stick or Mouse	Turret Rotator
Elevate Barrel	N/A	Look & Aim with Right Stick or Mouse	Barrel Elevator

Actors from Multiple Meshes

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Import the tank in 4 static mesh parts
- Assemble the parts using sockets
- Create our Tank_BP and test.

Import the Barrel

- Import the .fbx file
- Create a socket on the turret (eyeball position)
- Create a child of turret in Tank_BP
- Link to the barrel mesh and set socket
- Check it all looks and rotates ok.

Configuring a Tank

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Add mass to the tank
- Fine-tune track position
- Replace root component in Tank_BP
- Enable physics and assign a mass
- Set the tank as the Default Pawn
- Setup PlayerStart and debug start collisions.

Set Tank_BP as Default Pawn

- Create a BattleTankGameMode_BP
- Assign the tank as the Default Pawn
- Check the correct tank spawns on play.

3rd Person Camera Control

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Horizontal Coordinate System*
- Setup a Camera Spring Arm
- Why the Spring Arm alone isn't enough
- How rotations don't "commute"
- Binding mouse and gamepad to camera control.

*https://en.wikipedia.org/wiki/Horizontal_coordinate_system

Bind AimElevation in Blueprint

- Use the AimAzimuth as a template
- Setup corresponding binding for elevation
- Test it works

Fixing 3rd Person Camera Rotation

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

Setup Your Camera

- Use a Scene Root as azimuth gimbal
- Use the Spring Arm for elevation control
- Adjust the Spring Arm length
- Set the camera rotation to 0 (down the arm)
- Decide if you want the camera to roll or not.

User Interface (UI) in Unreal

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Create a Widget Blueprint for the aim point
- Decide the Player Controller with create the UI
- Create widget and add to viewport in Blueprint
- Override the Player Controller in the game mode.

Set the Player Controller

- Find the game mode blueprint for this level
- Set the Player Controller to be our new BP class
- Test the aiming dot shows up.

Main Menu Screens

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Create a dedicated Scene for the Main Menu
- Use the Level Blueprint to configure UI
- Add a background image to get started.

Add Widget from Level Blueprint

- Open the MainMenu Level Blueprint
- Create Blueprint to add widget to screen
- Test this works with a button, image or something
- Extra credit: get the mouse cursor showing.

UI Scale Box, Buttons & Mouse

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Show mouse cursor in Unreal UI
- Use a Scale Box for background image scaling
- Add a Start button
- Customise fonts inside our UI Widget
- Set anchors so UI scales to different aspect ratios.

Lay Out Your Main Menu

- Choose a font for your title text
- Consider adding a sub-title
- Add and style a Start button
- Check the scaling works well.

Controller Ready Navigation

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Bind Start button event to Blueprint
- Create custom WidgetReady event
- Make Start menu button focused on play
- Ensure we can quit from the game
- Aim towards Steam “Full Controller Support”.

Make Quit Button Work

- Ensure you have a Quit action bound
- Make it call the Quit event in BP.

Trial Packaging Your Game

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Creating a stand-alone game
- Setting the first level that loads
- Making sure the input mode works
- Setting-up for “Full Controller Support”.

Package for your OS

- Set the Editor and **Game** start maps
- Build for your platform
- Test it runs, try rescaling, try gamepad only.

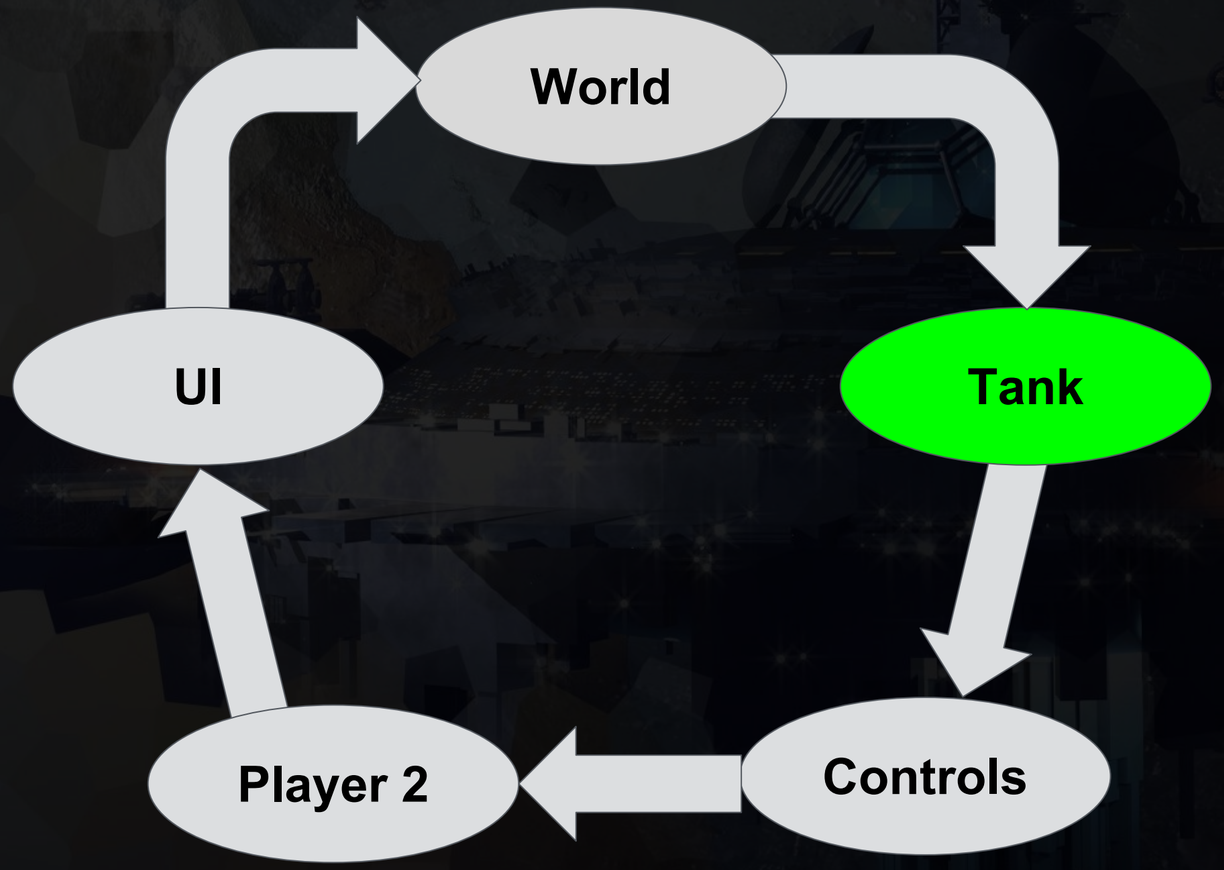
Delegating to Components

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

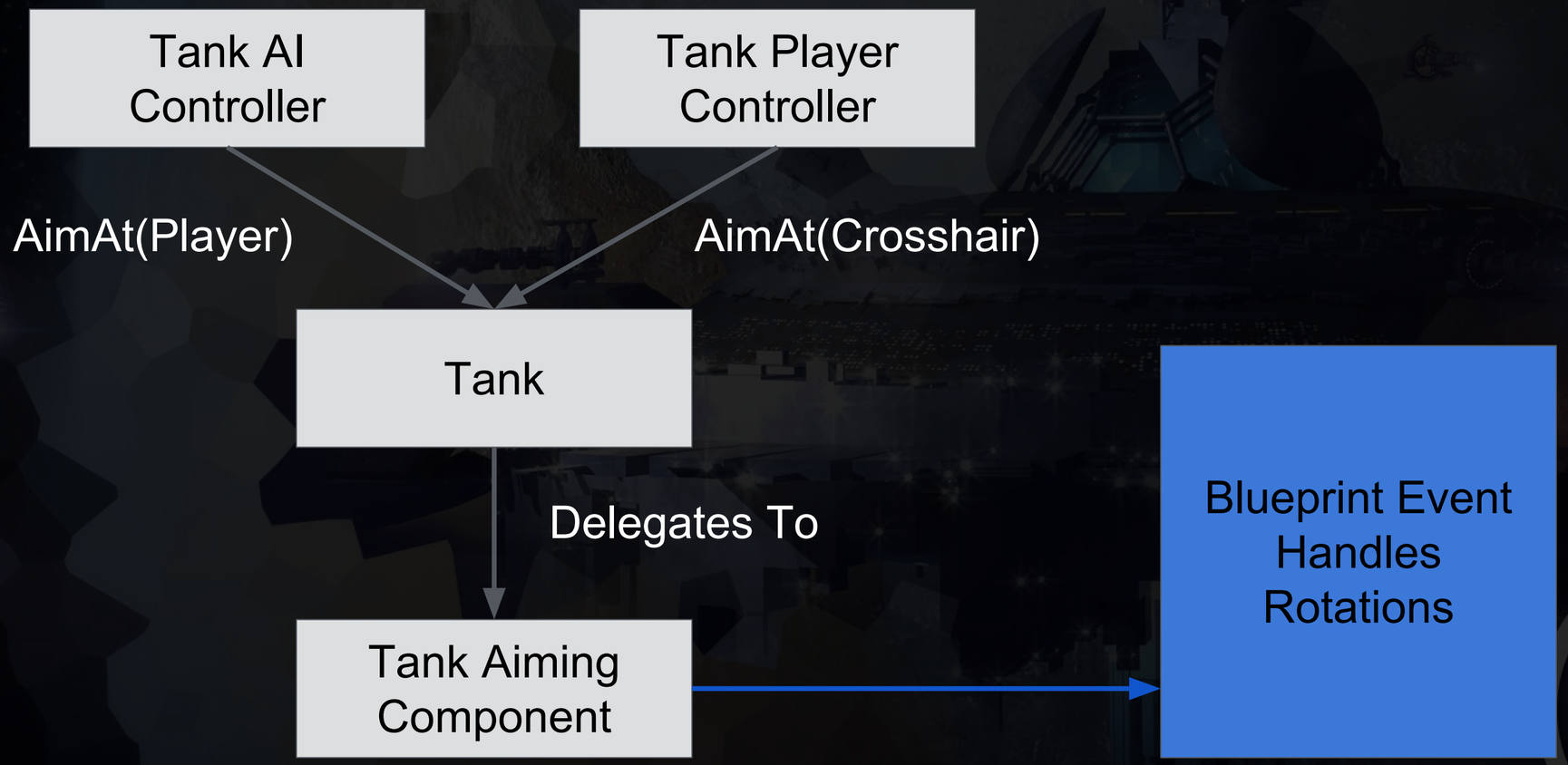
In This Video…

- How delegation can hide information
- Creating a custom Player Controller class
- Re-parenting Blueprint classes onto our C++

Our Iterative Cycle



Aiming Architecture



Create TankPlayerController

- Create a C++ class `TankPlayerController`
- Re-parent the TankPlayerController_BP onto this
- Good luck!

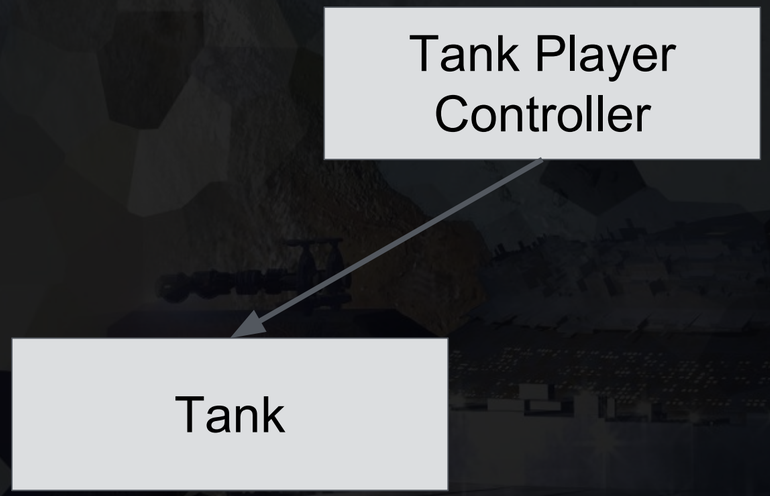
Using `virtual` and `override`

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- A `virtual` method can be overridden by children
- The `override` keyword is a sanity check
- Use `Super::` to include parents' functionality
- Use this to add `BeginPlay()` to PlayerController.

Aiming Architecture



Overriding in a Nutshell

- Children inherit their parents' methods
- To check you're actually overriding use `override`
- If the method was declared `virtual`, it can be overridden by any ancestor in the future.
- Specifying `virtual` on an already virtual method has no effect.

How to Override a Virtual Method

1. Find the signature in the API reference*
2. Copy the signature, postfix `override` in the .h
3. Define in the .cpp `without` virtual or `override`
4. Usually call `Super::` in the first line of code.

*https://docs.unrealengine.com/latest/INT/API/Runtime/Engine/GameFramework/APlayerController/index.html

Log Out Possessed Tank

- Remember your UE_LOG macro
- Consider an intermediate variable
- Remember to use * (contents at) for strings

Overloading and Polymorphism

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- What is polymorphism?
- How to overload functions.
- Using run-time polymorphism.

Polymorphism

- Ad-hoc - A different implementation per signature.
- Subtype - Parent pointers pointing at children instances.
- Parametric - One implementation, multiple signatures.

Polymorphism and C++

Polymorphism	Also known as...	Language Features
Ad-hoc		Overloading
Subtype	runtime polymorphism	Sub-classes and Virtual functions
Parametric	compile-time polymorphism	Templates

Make the Animals Behave

- Each call should call the right behaviour.
- What would happen if this wasn’t a pointer?

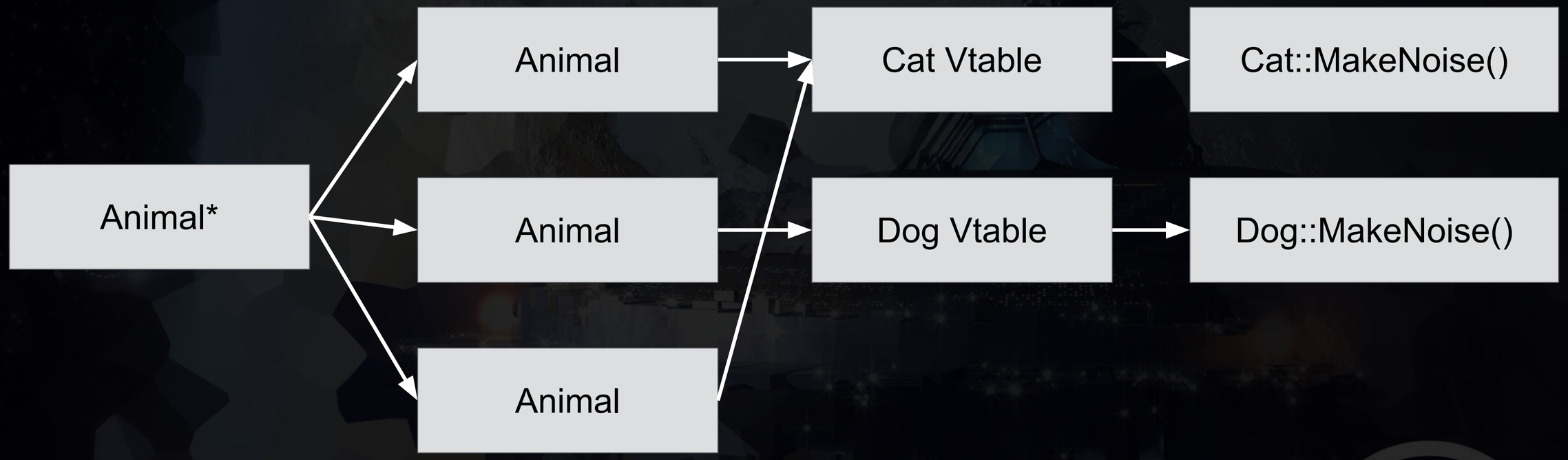
Virtual Functions and Vtables

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- How methods are called.
- Why we need the virtual method.
- How Vtables implement this.
- See how the assembly changes.

Vtables


Animal*
Animal
Animal
Animal
Cat Vtable
Dog Vtable
Cat::MakeNoise()
Dog::MakeNoise()

Creating an AI Controller Class

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- How to create a `AIController` based C++ class
- Assigning an AI Controller to a Pawn
- Verifying which pawns are possessed
- Logging possession to the console.

Aiming Architecture

Log the Possessed Tank

- Use similar code to our PlayerController
- Try placing some more tanks in the level
- You should see one log entry for each tank.

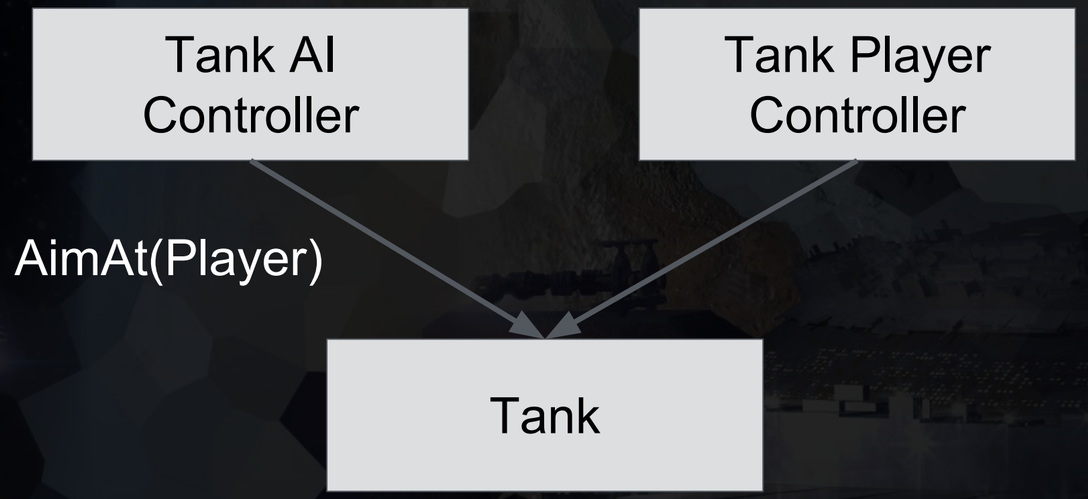
Get the Player Controller with C++

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

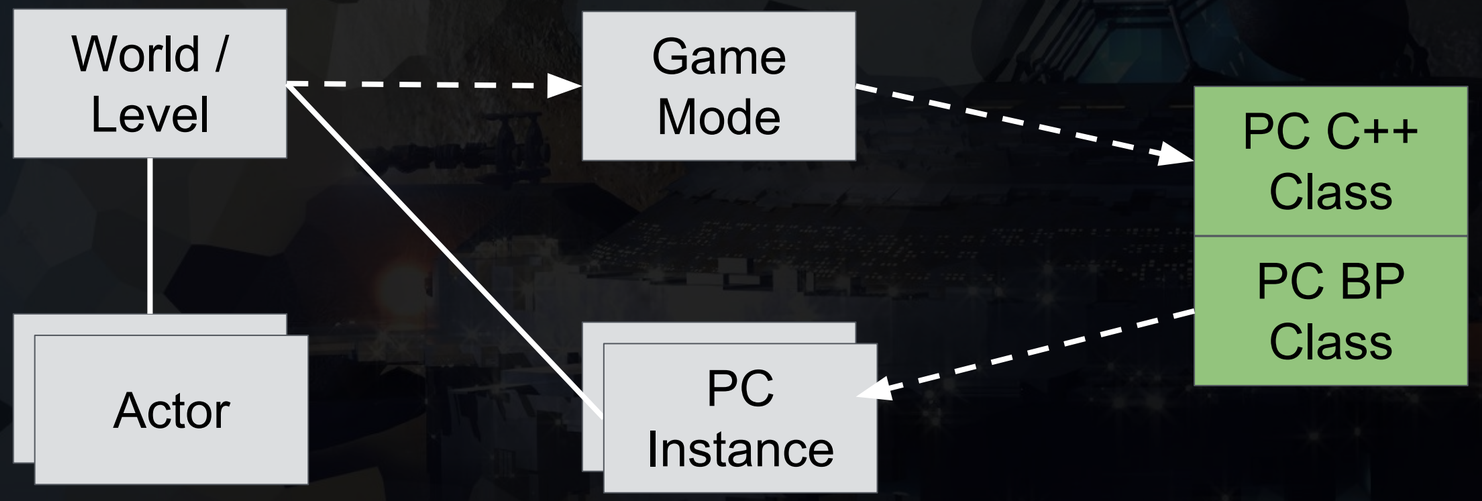
In This Video…

- Getting the AI to find the player position
- We won't implement line-of-sight for simplicity
- `UGameplayStatics::GetPlayerController()`
- Or `GetWorld()->GetFirstPlayerController()`

Aiming Architecture



Finding the Player Controller



PC = Player Controller

Get the Player Tank

- `GetWorld()->GetFirstPlayerController()`
- Protect your pointer
- Cast to a `Tank`
- Log on `BeginPlay()` to check it worked.

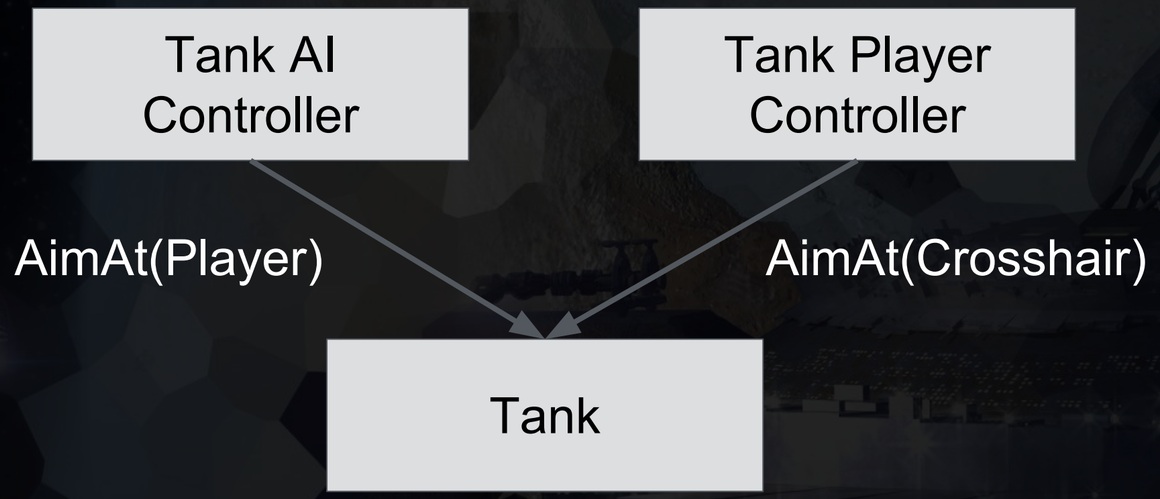
Add `Tick()` to `PlayerController`

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Revise adding engine methods into new classes
- Pseudocode our initial aiming logic
- Learn about Visual Assist for Visual Studio.

Aiming Architecture



Make the Player Controller Tick

- Declare and define the `Tick()` method
- Check it works with a log message.

Creating an Out Parameter Method

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Out parameters smell a little but are used a lot
- Allows you to return a `bool` and a `FVector`
- Alternative architecture would be a struct or class
- We'll do it this way to get you more comfortable with creating your own methods using out parameters.

Write GetSightRayHitLocation()

- Return a bool, and make the method `const`
- Return `FVector HitLocation` as out parameter
- Remember you will need `&`, and think why
- Bonus: pseudocode actual functionality
- Super bonus: code working functionality!

Finding Screen Pixel Coordinates

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Use `FVector2D()` to store pixel coordinates
- This is two floats, pixels can be non-integer
- Revising `UPROPERTY(EditAnywhere)` and more.

Find `ScreenLocation`

- Create `CrossHairXLocation` and set to 0.5
- Create `CrossHairYLocation` and set to 0.33333
- Make both of these `EditAnywhere` for BP
- Calculate an `FVector2D ScreenLocation`
- This will be a pair of floats specifying the pixel coordinates that we'll want to de-project later.

Using `DeprojectScreenToWorld`

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- How to find the camera look direction
- What the `WorldLocation` parameter does
- `WorldDirection` returned is a unit vector.

Do the “De-Projection”

- `DeprojectScreenPositionToWorld`
- Lookup the method signature online
- Log the `WorldDirection` it returns
- Ignore the `WorldLocation` it returns, it’s the camera’s World Location but isn’t needed.

Using `LineTraceSingleByChannel()`

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- We want world position of anything visible
- `GetWorld()->LineTraceSingleByChannel()`
- Use the `ECC_Visibility` channel for what's seen
- Remember `HitResult` is a rich object
- Use `HitResult.Location` for Location member.

Write `GetLookVectorHitLocation()`

- Make a `LineTraceRange` parameter = 10km
- Line-trace out to this distance in `LookDirection`
- Trace by ECC_Visibility channel (if you can see it)
- Return `true` if hits, and `HitLocation` as OUT
- Return `false` if nothing is hit
- Log-out to see what you're hitting.

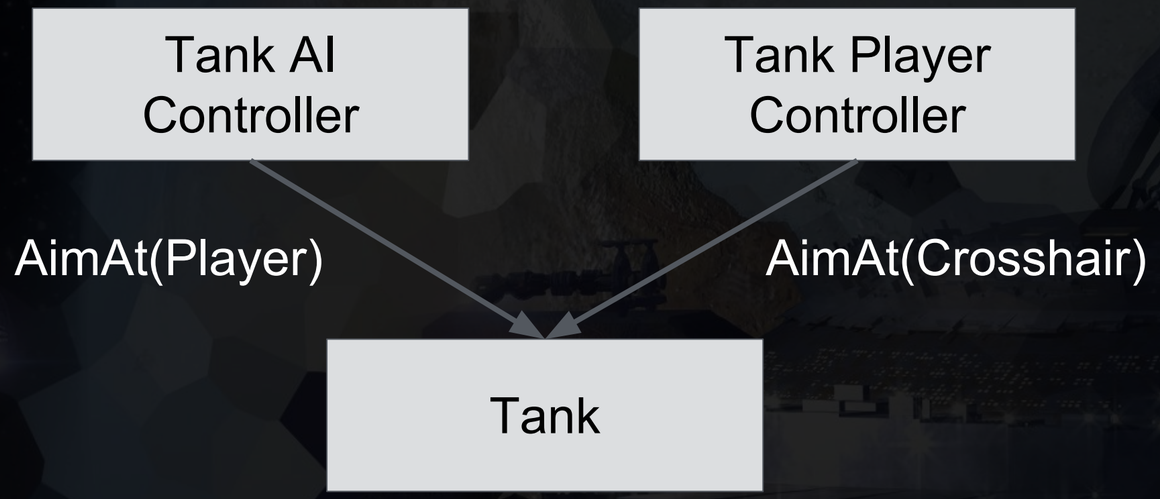
Unify Player & AI Aiming

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- AI and Player possessed tanks aim the same way
- Later the tank will delegate aiming
- But the AI/Player controllers don’t care
- This provides nice abstraction
- We also hide implementation details
- … and make the game more fair.

Aiming Architecture



Get AI Tank Reporting Aim

- Add Tick() into TankAIController class
- Call `Tank->AimAt()` every frame
- You should get 3 logs per frame now.

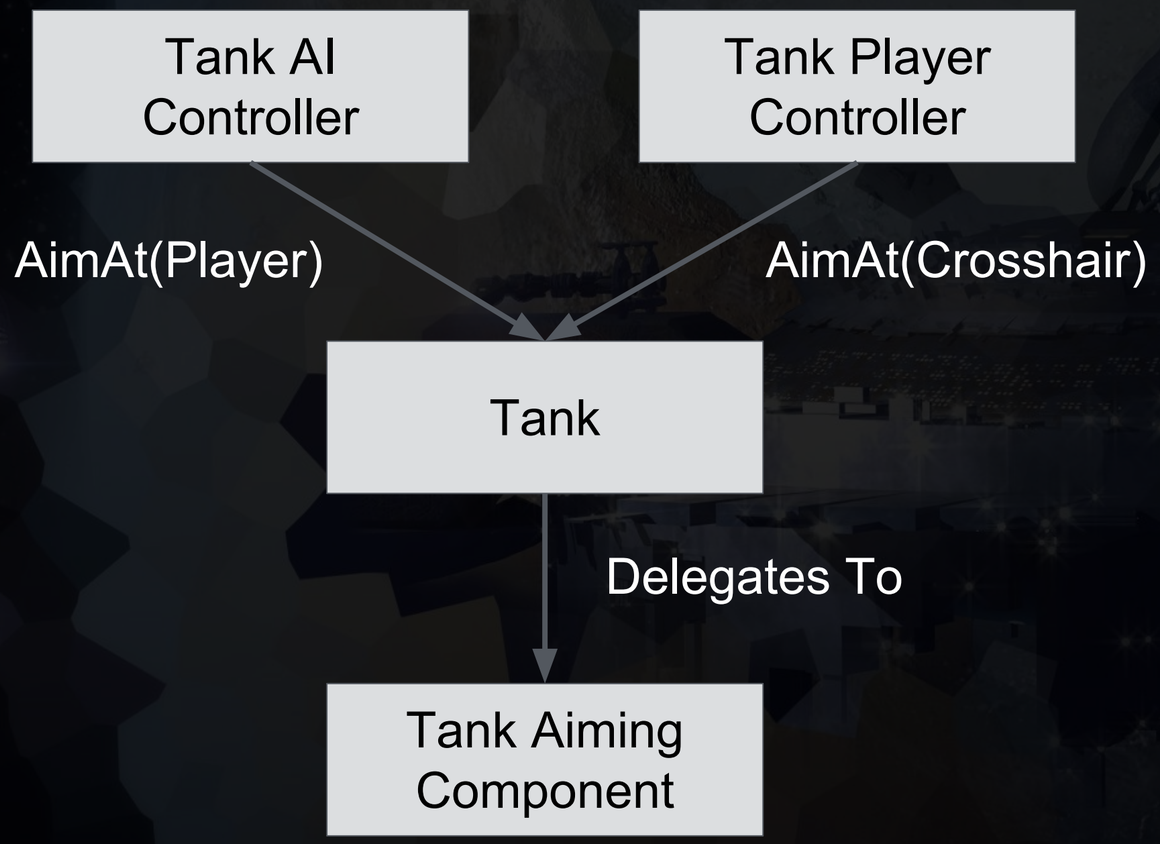
Create Default Sub Objects in C++

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- You can add required components in C++
- Our Tank Aiming Component is a good candidate
- We will delegate all `AimAt()` requests…
- … regardless of their source (AI or player).

Aiming Architecture



Move Aim Logs to Component

- **Move** the aim logging to the new component
- Test that the message gets through
- Ensure you’re comfortable with what’s happening.

BlueprintCallable()

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Why `StaticMeshComponet` is prefixed with `U`
- Creating a setter for the `barrel` reference
- How to name parameters in setters
- Using `BlueprintCallable()` to call C++ from BP
- Finding the start position of or projectile.

Our Projectile Flight

SuggestProjectileVelocity()

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- How speed and velocity relate
- The high and low projectile arc
- Setting a launch speed on the tank
- Introducing `SuggestProjectileVelocity()`

Projectile Speed & Velocity

Two Projectile Arcs

https://en.wikipedia.org/wiki/Trajectory_of_a_projectile

Research SuggestProjectileVelocity()

- Research the method
- Recap this video
- Make sure it all makes sense.

Predict Projectile Landing Point

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Use `SuggestProjectileVelocity()` in Unreal
- Work out where a projectile will land.

Get Launch Direction

- Use `SuggestProjectileVelocity()`
- Use `OutLaunchVelocity.GetSafeNormal()`
- Log out the required aim direction.

Using FRotators in Unreal

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- A `FRotaor` is a struct
- It contains Roll, Pitch and Yaw as floats
- Convert using `.Rotation()` method
- Report aim direction as a rotator
- Log result to the console in Unreal.

Pseudocode `MoveBarrel()`

- Abstract at higher level than actual code
- Use plain English
- Have fun, it’s just an exercise.

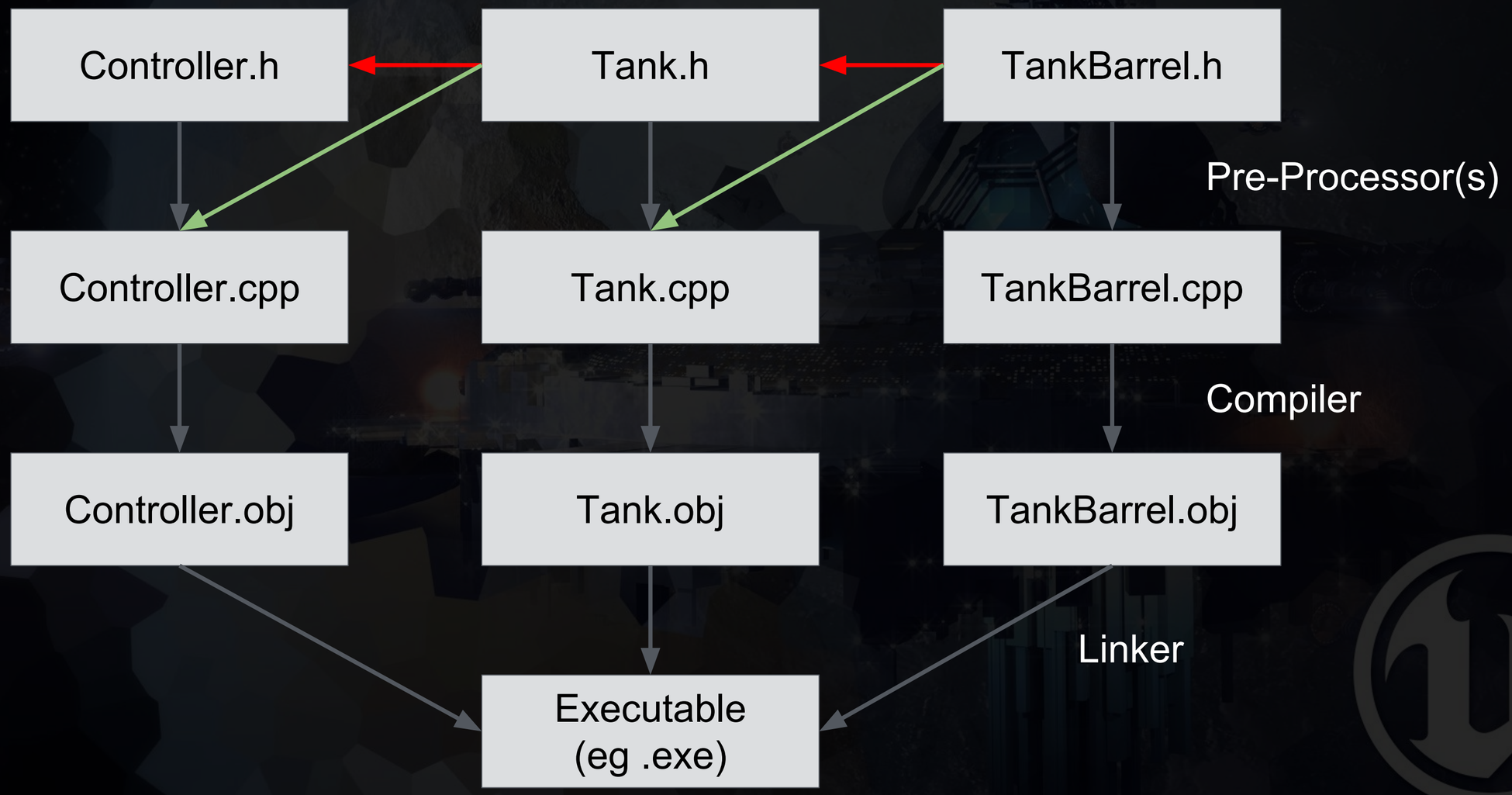
The C++ Compilation Process

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

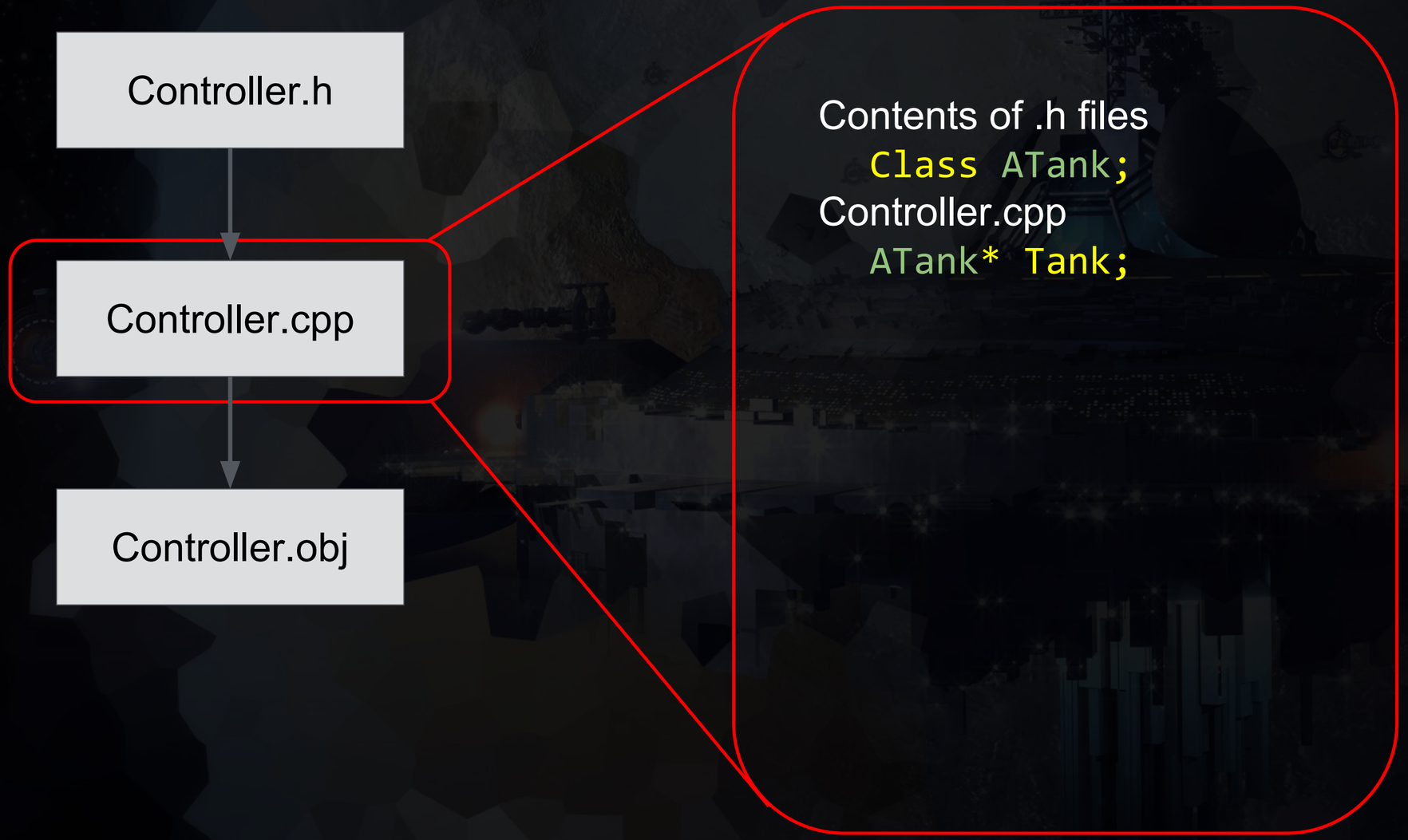
In This Video…

- More about the Unreal Header Tool (UHT)
- Pre-processing happens first, e.g. on macros
- Then compilation produces .obj files
- These .obj files are linked by the linker
- How to `#include` strategically.

The Compilation Process



The Compilation Process



Compile Pass

Try This For Yourself

- Open TankPlayerController.h
- Move `#include “Tank.h”` to the .cpp file
- Try it both above and below the following line…
- `#include “TankPlayerController.h”`
- Put your code back to original state for now.

Using Forward Declarations

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Usually only include classes in `.h` when inheriting*
- But you will need referenced types to compile
- You *could* `#include` other in the .cpp above
- We don't want order dependence
- So forward declarations are the way to go.

** or when not a pointer, or a few other cases.*

BlueprintSpawnableComponent

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- In actor blueprints you have custom components
- Static mesh components don’t appear by default
- Use `BlueprintSpawnableComponent` annotation
- Using `hidecategories = ("CategoryName")`

Add Remaining Properties

- Add a max and min elevation in degrees
- Give sensible defaults in C++
- Prevent self-collision.

Review Our Execution Flow

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- How to disable or enable tick on various classes
- `GetWorld()->GetTimeSeconds()` for logging
- Documenting your execution flow for clarity
- Change parameter names for clarity.

How The Barrel Elevates

How to Report Bugs

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- If something's weird break it down
- Use logs or the debugger to follow each step
- `SuggestProjectileVelocity()` has a bug*
- … it MUST have an optional parameter!?
- Moving to forward declarations.

https://answers.unrealengine.com/spaces/11/bugs-and-crashes.html

Move to Forward Declarations

- Report that bug to Epic if you want
- Re-cap the Forward Declarations video if required
- Move to forward declarations in all .h files
- Test your game still runs.

Using `Clamp()` to Limit Values

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- `FMath::Clamp<type>(Input, Min, Max);`
- Very useful for restricting value ranges
- Clamp our Barrel's elevation
- Wire it to the aiming component
- Test barrel elevation works.

Clamp The Elevation

- Use `FMath::Clamp()`
- Test this does indeed restrict elevation.

CHALLENGE - Turret Rotation

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

This mid-section challenge will help you integrate your knowledge and really cement what you've done in the past few lectures. It will also give you a great foundation of practical understanding on which to build. Please give it a good shot before watching my solution.

Get the Turret Rotating!

- Revise the last few lectures if required
- At least take a look at your commits*
- Work through the Blueprints and C++ code
- See if you can get the turret rotating
- Don’t worry how that barrel moves.

**You have been committing to source control right ;-)*

CHALLENGE - Turret Rotation Pt.2

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

This is the 2nd part of the solution to this section's longer challenge. We'll be finishing off the turret rotation, giving us complete barrel aiming control by the end :-)

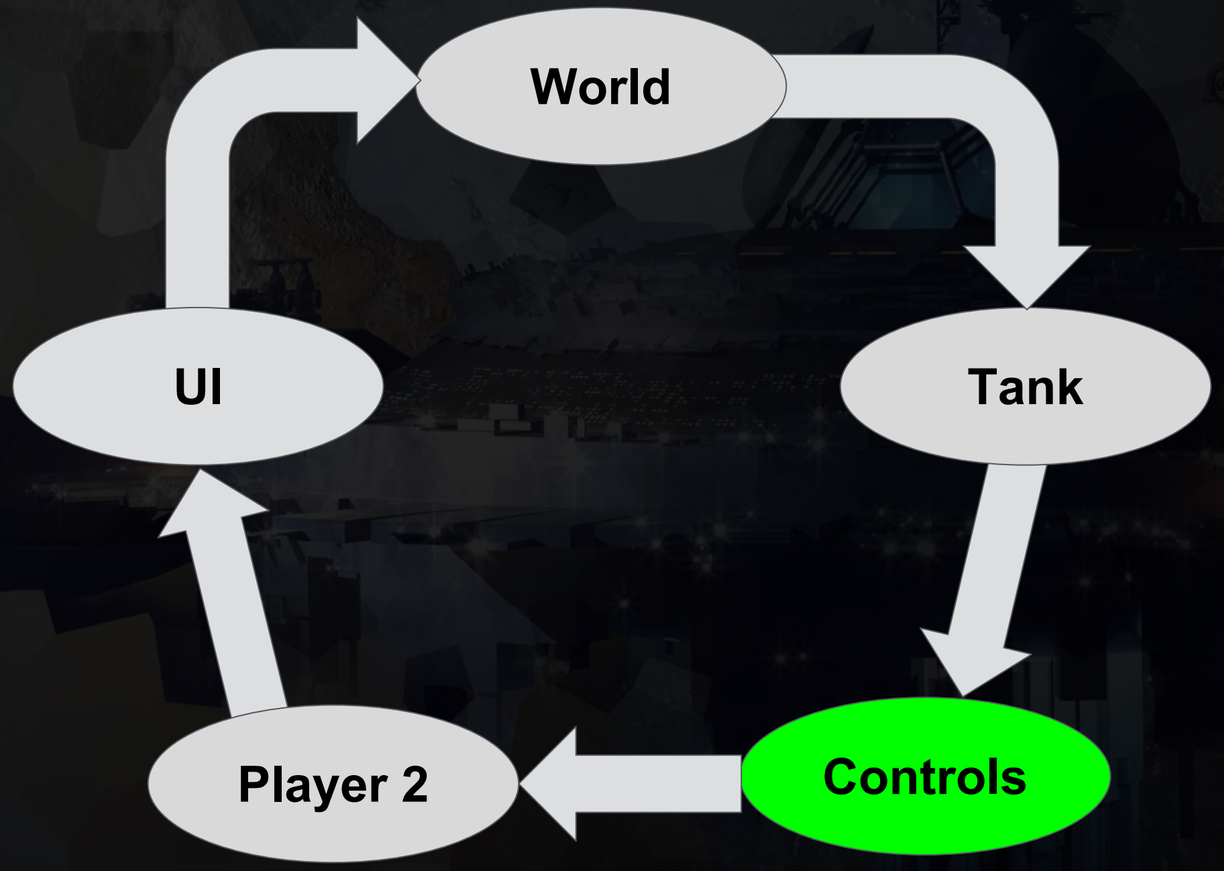
Setting Up Projectiles

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Create a public `Fire()` method on our tank
- Bind input via Blueprint
- Call this new C++ method to test
- Create a `Projectile` class, and Blueprint it.

Our Iterative Cycle



Setup the Firing Code

- Create a `Fire()` method on the tank
- Make it `BlueprintCallable`
- Bind input, and wire-up in Blueprint
- Test by logging to console from C++.

Upgrading to Unreal 4.12

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Multiple versions of the engine take up GB
- Upgrade Building Escape and Battle Tank
- Learn more about using source control
- Using Stash in source control
- Fixing issue with overlapping collision volumes.

Upgrade Both Projects to 4.12

- Upgrade Building Escape, test & commit
- Upgrade BattleTank, test & commit
- Remove previous version(s) of Unreal
- Consider deleting other Vault local content.

4.12

Working Round Awkward Bugs

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- About AutoWeld compound objects
- Working through self-collision issues
- Disabling gravity on subobjects
- A reminder Unreal is designed for humanoids.

Find the Remaining Problem

- What other meshes could be responsible?
- Go through a similar process
- See if you can solve the bug.

4.12

Using SpawnActor<>() to Spawn

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Using `TSubclassOf<Type>`
- More about forward declarations
- How to use `GetWorld()->SpawnActor()`
- How to spawn projectiles from a weapon.

Spawn the Projectile

- Fill-in the `SpawnActor()` parameters
- Test that it works.

Lecture Resources

01. Community Discussions (for intros, shares, discussions, tips etc)
02. https://docs.unrealengine.com/latest/INT/Programming/UnrealArchitecture/TSubclassOf

4.12

Projectile Movement Components

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Recap use of `CreateDefaultSubobject()`
- Use a `ProjectileMovementComponent`
- Get our tank delegating launch to projectile.

Add a Default Sub Object

- Add the projectile movement component in C++
- Set it’s `bAutoActivate` property to false
- Test that your tank now comes with one
- Also ensure you can’t edit properties.

Lecture Resources

01. Community Discussions (for intros, shares, discussions, tips etc)
02. https://github.com/UnrealCourse/04_BattleTank/commits/master

4.12

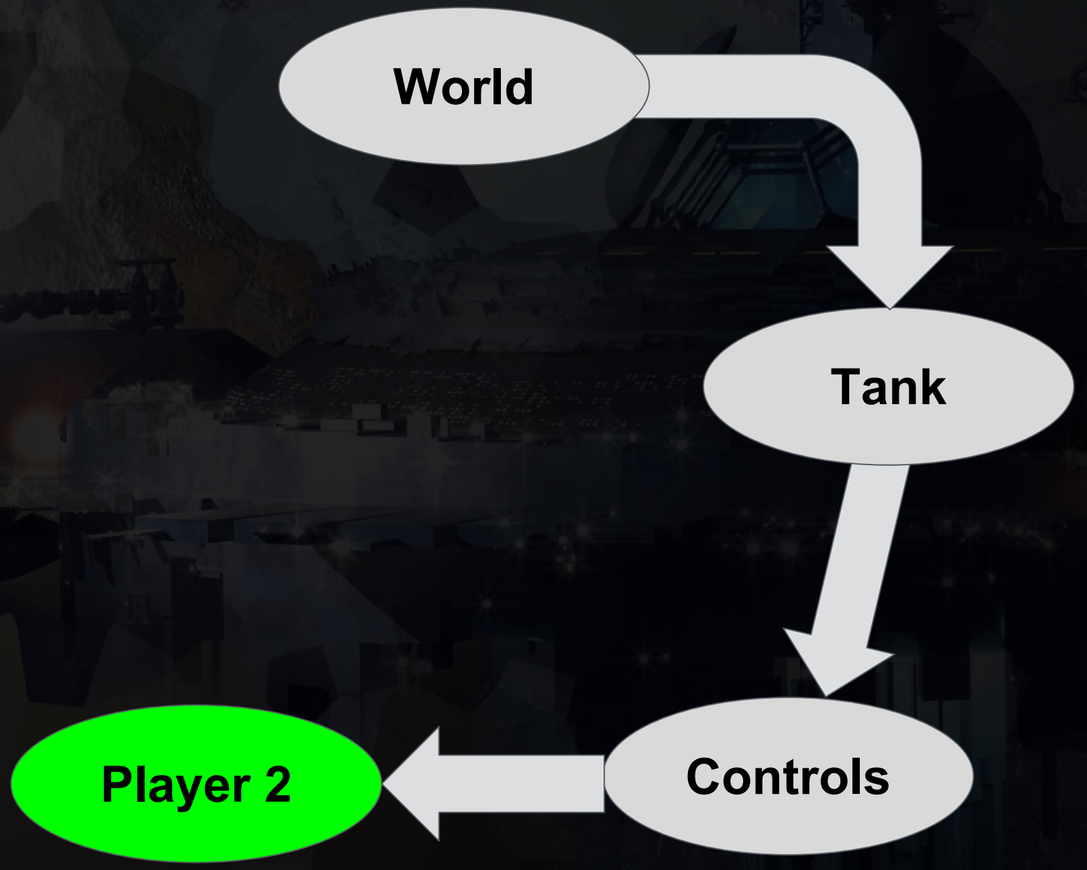
Making AI Tanks Fire

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Inline some code for readability
- Inlining can also be called “defactoring”
- Less lines of code is often better*
- * everything else being equal
- `FPlatformTime::Seconds()` is an accurate timer
- Make AI tanks fire on every frame.

Our Iterative Cycle



Inline the Two Private Getters

- Remove the two methods
- This is called in-lining, or “de-factoring”
- Put the code in the `Tick()` method
- You can remove code in `BeginPlay()`.

EditAnywhere vs EditDefaultsOnly

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- `EditAnywhere` allows all instances to be edited
- For example each AI tank could be different
- `EditDefaultsOnly` allows “architype” editing
- In other words, all tanks must be the same
- Think which you want in future.

Question every `EditAnywhere`

- Search the project
- For each instance decide what you want
- Make the changes and test.

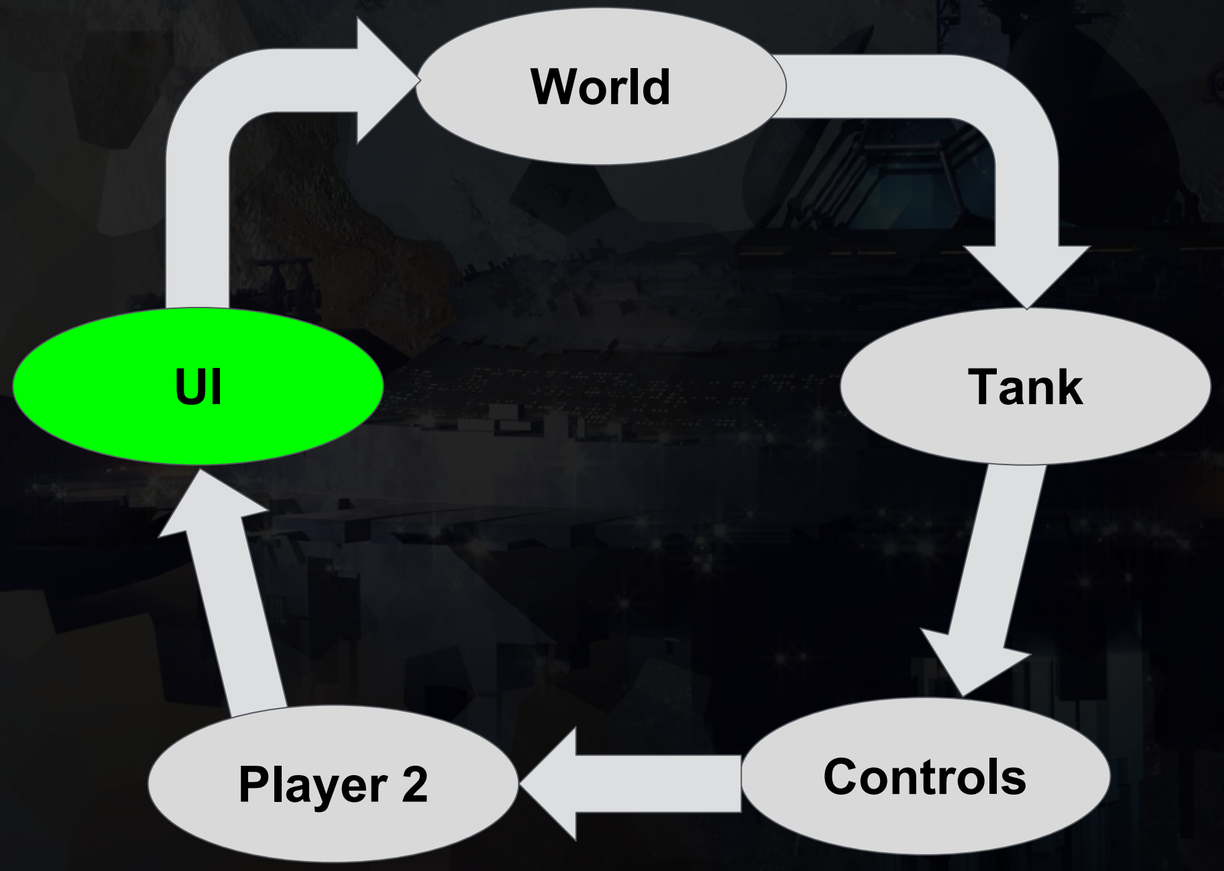
Adding a Quit Button

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Using primitive colliders in Unreal
- Adding a quit button to our main menu.

Our Iterative Cycle



Add a Quit Button

- Add a Quit Button to the main menu
- Call application quit from Blueprint
- Test it stops play.

4.12

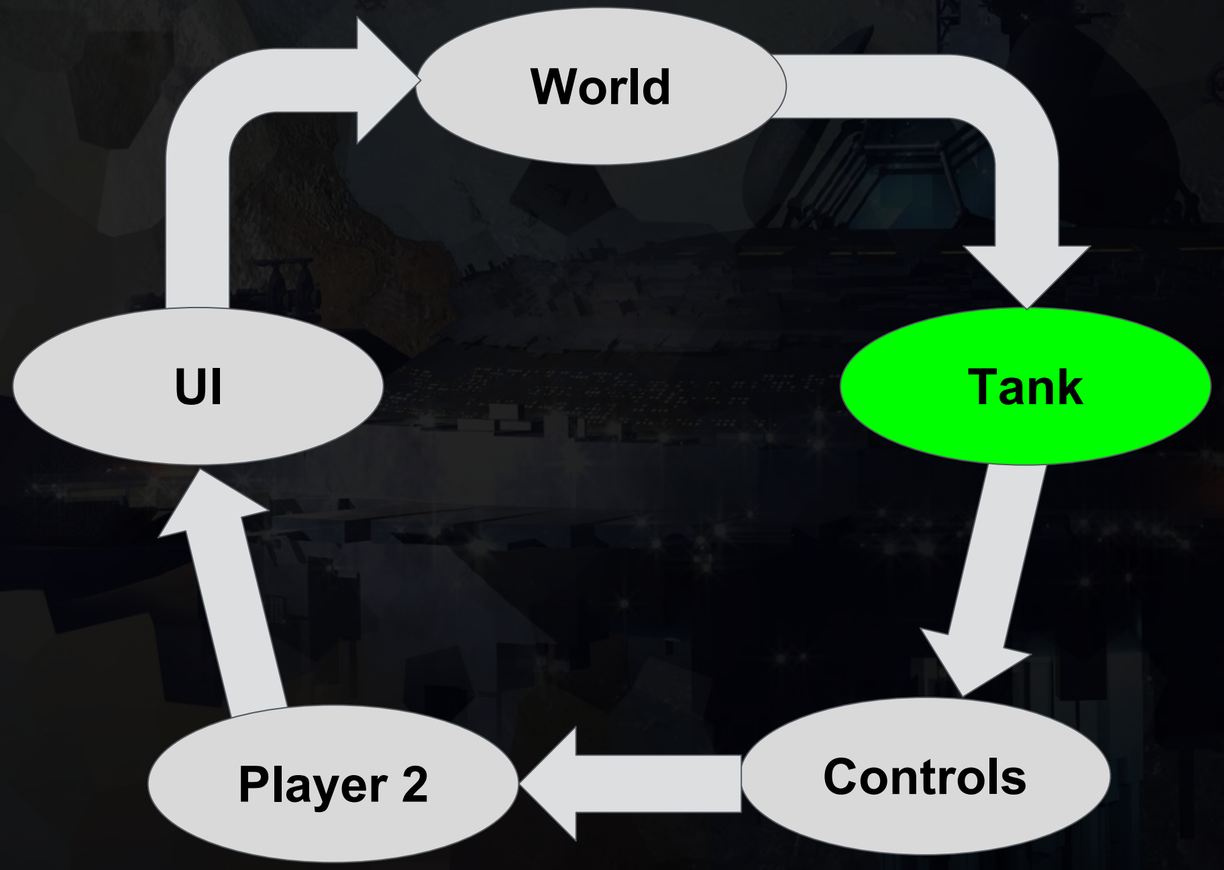
Setup Track Throttles

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Base Tank Tracks on `UStaticMeshComponent`
- Create a `BlueprintCallable` throttle method
- Bind input to track throttles
- Discuss what Input Axis Scale does.

Our Iterative Cycle



Base Tracks on C++ Class

- Create a `TankTrack` class
- Inherit from `UStaticMeshComponent`
- `BlueprintSpawnableComponent` as barrel / turret
- Replace tracks on Tank blueprint
- Test the game still works.

4.12.4

`ApplyForceAtLocation()` in Action

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- `GetComponentLocation()` does what it says!
- Find root: `GetOwner()->GetRootComponent());`
- Cast to `UPrimitiveComponent` so you can…
- `AddForceAtLocation();`
- Estimate sensible defaults for driving forces.

Suggest a Sensible Default

- Remember Force = mass * acceleration
- Be clear on what units we're using
- Guess an initial force assuming no friction
- Getting within a factor of 10 of my guess is OK!

Guessing Required Force

- Tank mass is 40,000 kg
- Force = mass x acceleration
- Imagine $10ms^{-2}$ (1g) acceleration*
- Force is therefore around 400,000 Newtons

** a real tank would be more like 2-3ms^{-2} but it's a game.*

Lecture Resources

01. Community Discussions (for intros, shares, discussions, tips etc)
02. 0-60 in 10s on Wolfram Alpha

4.12.4

Physics Materials & Friction

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- You can assign a physics material under collision
- Friction is combined between two surfaces
- The coefficient is the proportion of the contact force that can be exerted sideways before slip.
- Adjust friction and driving forces to get movement.

Tweak Your Physics Settings

- Set your track max driving force
- I’m using 40,000,000 Newtons
- Set your track physics material friction value
- I’m using 0.2 for this
- This means the friction is 0.2 x the contact force.

4.12.4

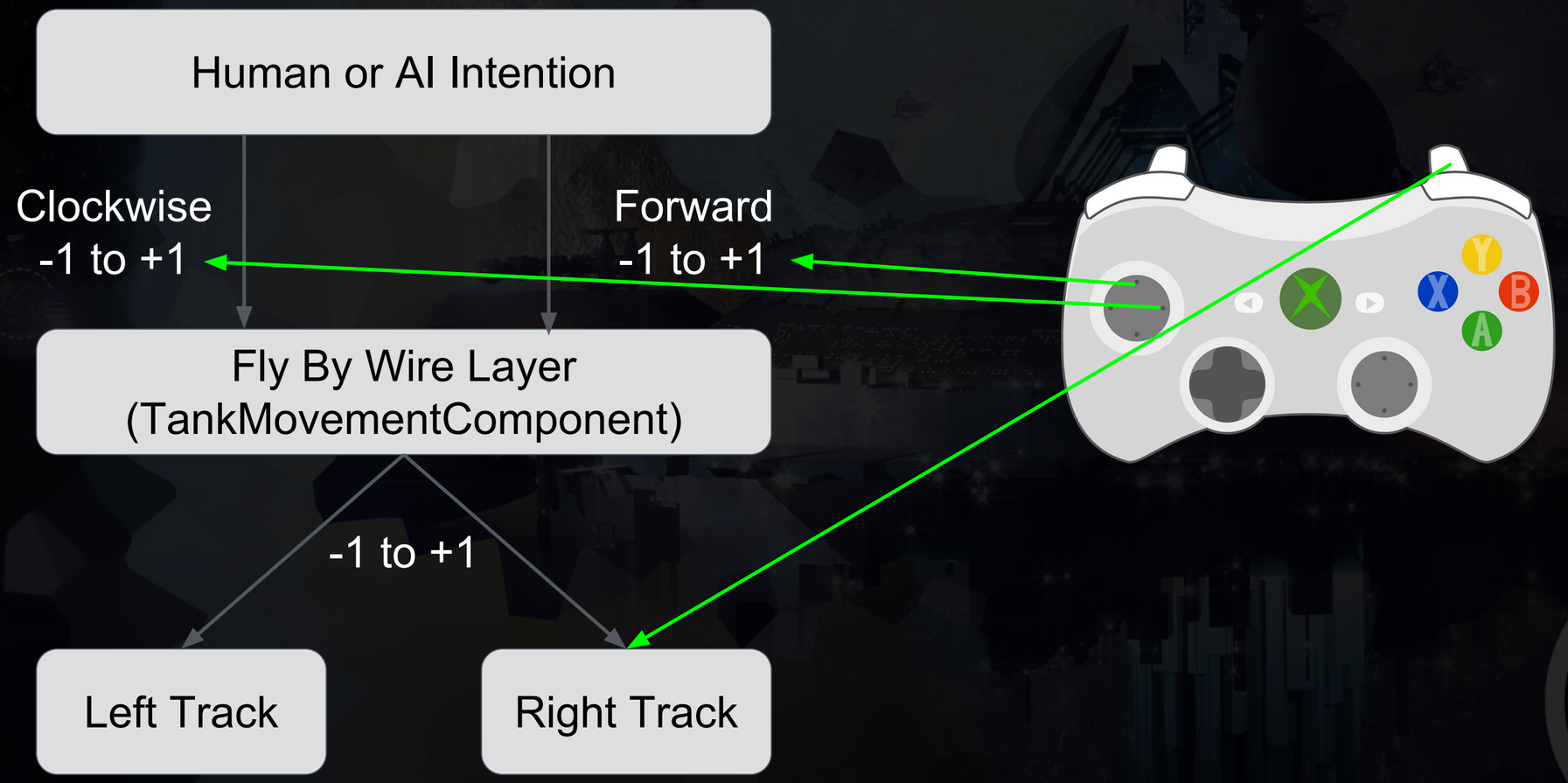
Fly-by-Wire Control System

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Fly-by-wire means translating control intention
- How control intention maps to track throttles
- Creating a `TankMovementComponent` C++ class
- Why inherit from `UNavMovementComponent`

Our Fly-by-Wire Architecture



4.12.4

Using `BlueprintReadOnly`

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Bind some input for forward and backward
- Make the method `BlueprintCallable`
- Make `TankMovementComponent` a default on tank
- Make a protected tank variable to store pointer
- Make this pointer `BlueprintReadOnly` pointer
- Test that you get a log of +/-1.

Wire-up `IntendMoveForward()`

- Bind some input for forward and backward
- Make the method `BlueprintCallable`
- Make `TankMovementComponent` a default on tank
- Make a protected tank variable to store pointer
- Make this pointer `BlueprintReadOnly` pointer
- Test that you get a log of +/-1.

Lecture Resources

01. Community Discussions (for intros, shares, discussions, tips etc)
02. Creating C++ Variables For Use In BP
03. BlueprintCallable docs

4.12.4

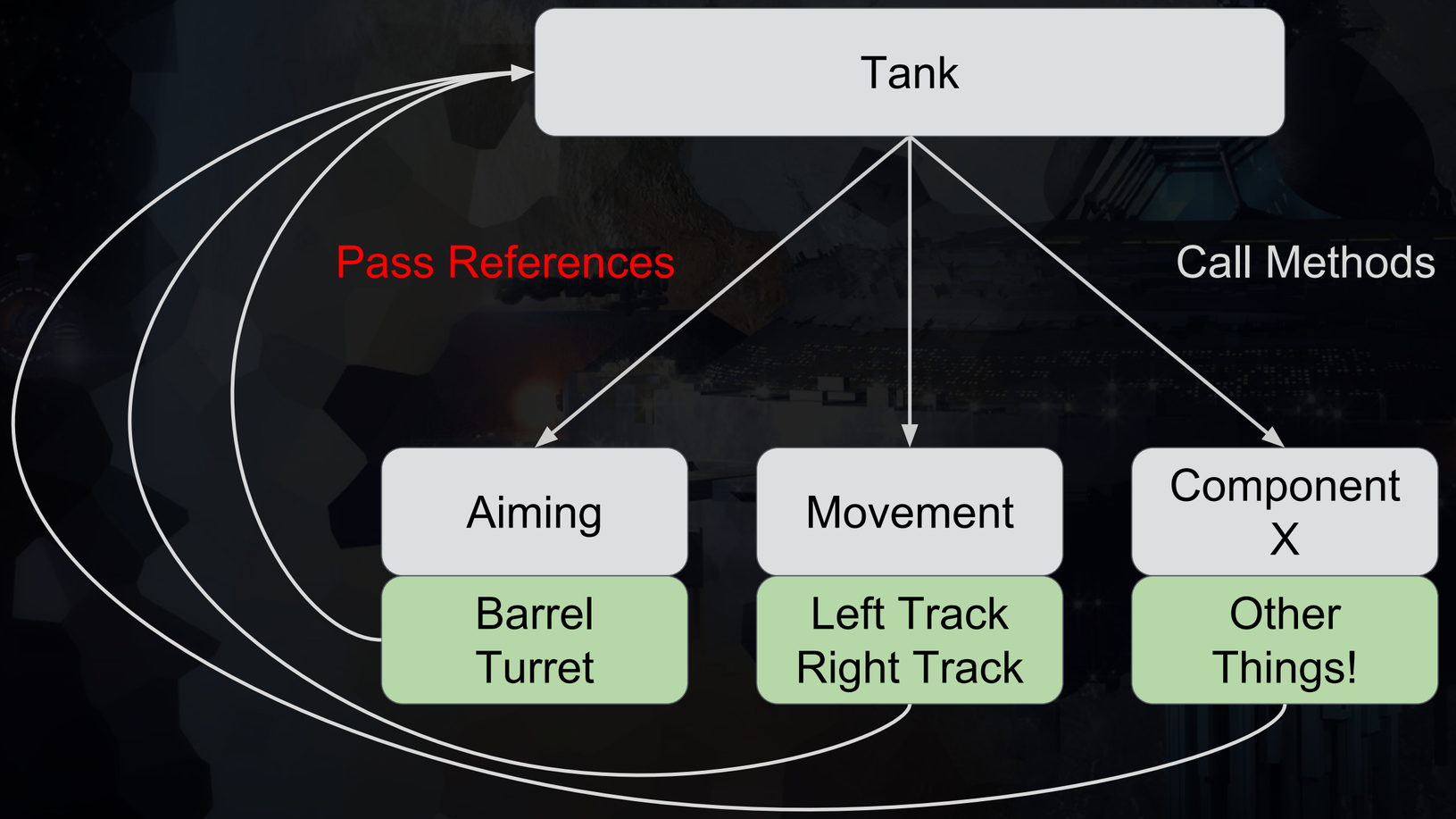
A Better Component Architecture

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Actor components require instance references
- We were passing these references from the tank
- But we could equally keep them locally
- Move to composing our actor in Blueprint
- Create an initialise method for aiming
- Test it works and hail the simpler code.

Our Component Architecture



Finish The Movement

- Finish `IntendMoveForward()`
- Make sure the tank moves
- Make sure the speed is variable.

4.12.4

Completing Manual Tank Movement

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Add `IntendTurnRight()` method
- Bind firing input to the “A button”
- Test we can move manually with fly-by-wire.

4.12.4

Introducing AI Pathfinding

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

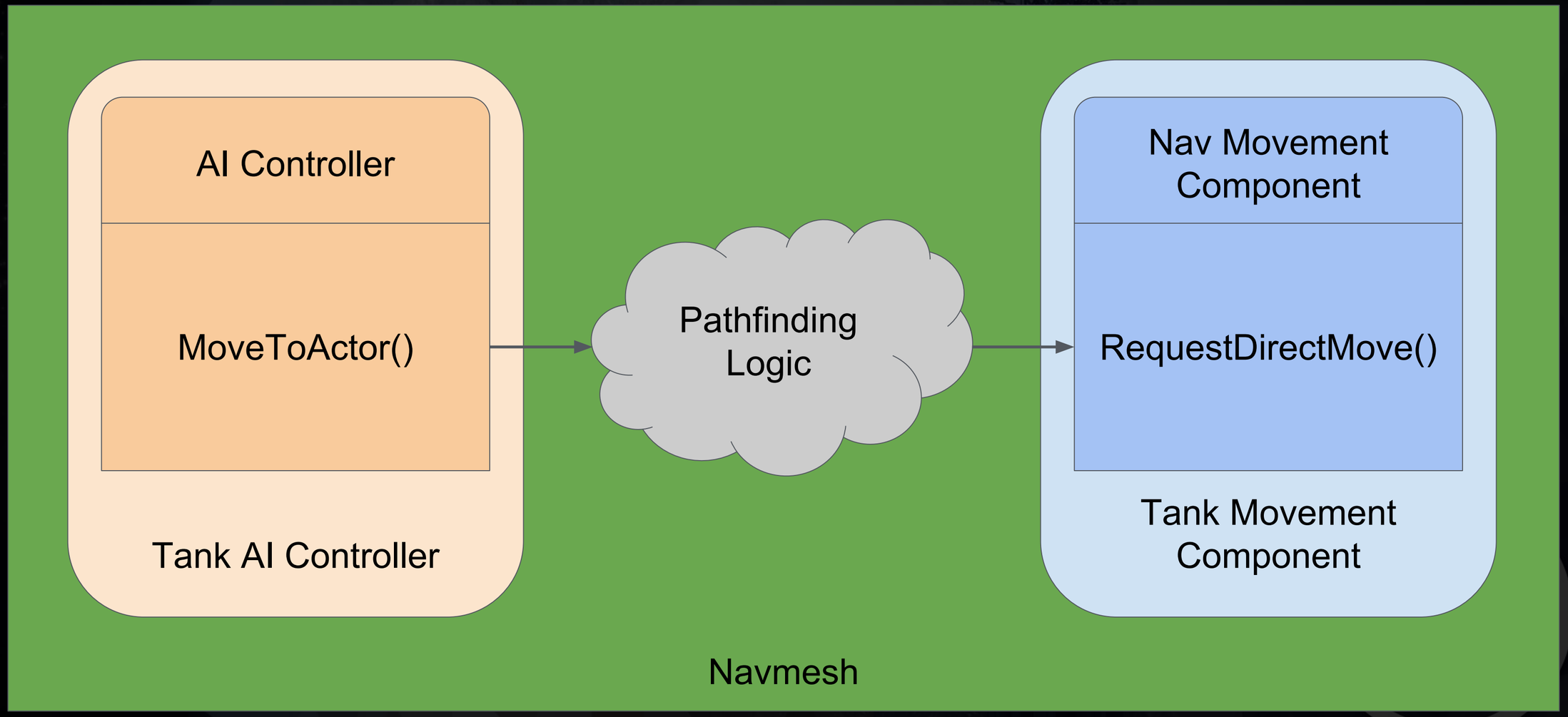
In This Video…

- Pathfinding is finding the shortest possible path
- This requires some (artificial) intelligence
- All pathfinding must happen on a navmesh
- Adding Nav Mesh Bounds to the level
- An overview of how `MoveToActor()` and `RequestDirectMove()` work.

Place Your Nav Mesh Bounds Volume

- Place a 100 x 100m bounds volume in the world
- Make sure you have obstruction(s) e.g. hills
- See the effect of raising and lowering the volume
- Leave Player Start and at least 1 AI tank are inside.

How We're Using Pathfinding



Lecture Resources

01. Community Discussions (for intros, shares, discussions, tips etc)
02. Unreal's Volumes Documentation
03. A* Pathfinding on Wikipedia

4.12.4

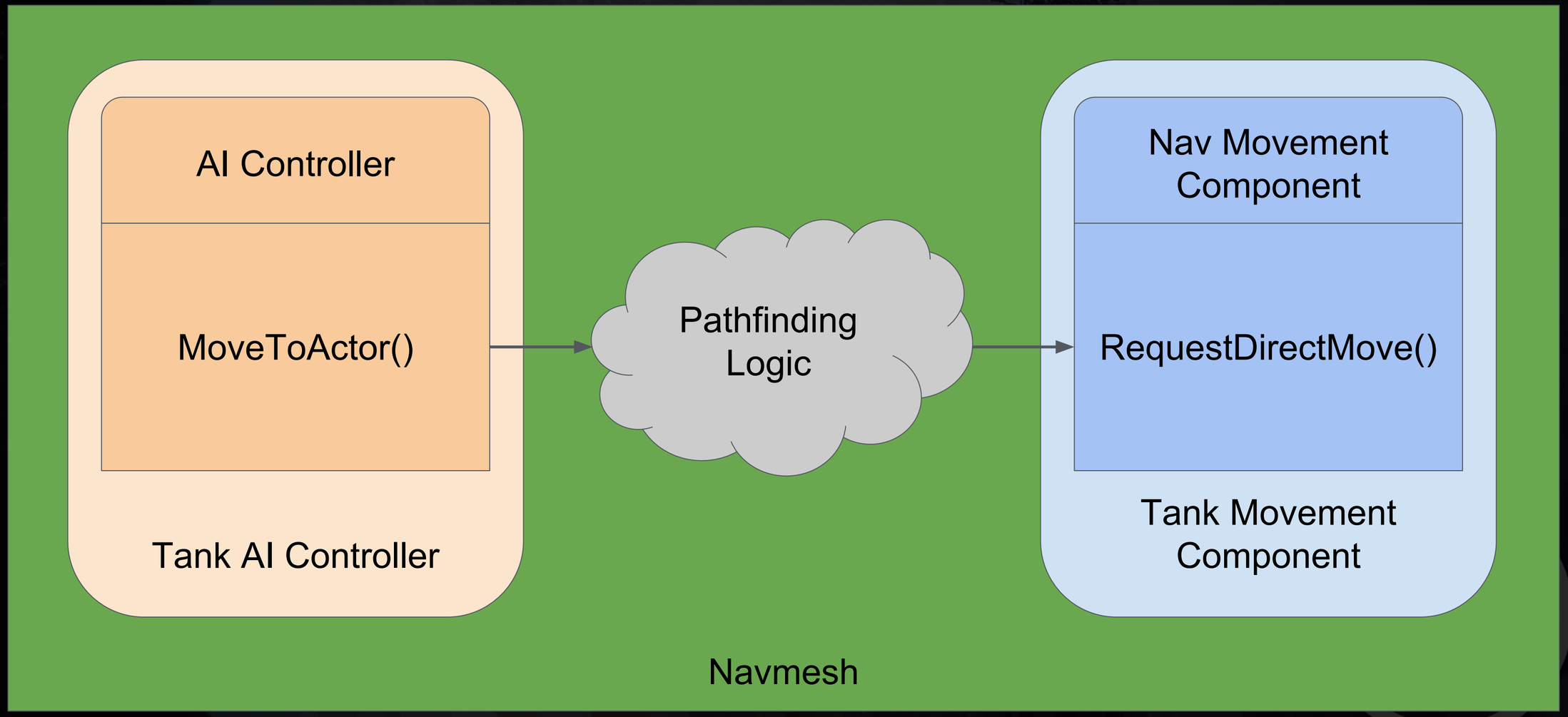
Dissecting `RequestDirectMove()`

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- We have access to Unreal's source code
- Let's look into the `UNavMovementComponent.h`
- We're looking for `RequestDirectMove()`
- We'll override it without calling `Super`
- We can then get the golden `MoveVelocity` vector
- AI tanks can now use our fly-by-wire controls!

How We're Using Pathfinding



Get `RequestDirectMove()` Signature

- Find the `NavMovementComponent.h` file*
- Copy the signature of `RequestDirectMove()`
- Override it in your `TankMovementComponent.h`
- No need to call Super this time, we're replacing
- Log the tank name and value of `MoveVelocity`.

***Engine > UE4 > Source > Runtime > Engine > Classes > GameFrameWork**

4.12.4

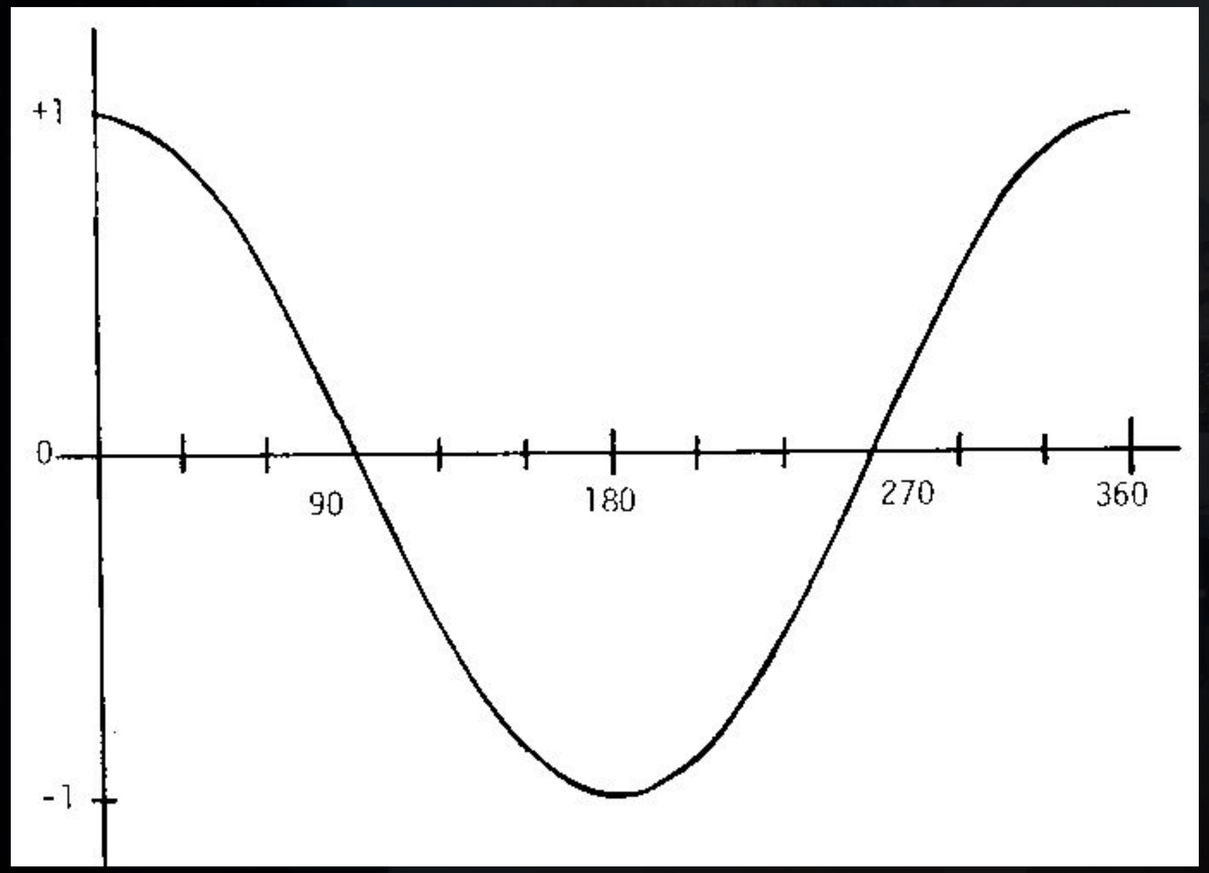
DotProduct() Vector Operator

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Focusing on controlling forward speed of AI
- If target in front, move forward full speed
- If target to side, don’t move forward
- Vary smoothly in-between
- This sounds like a cosine function to me!
- Using `FVector::DotProduct()`

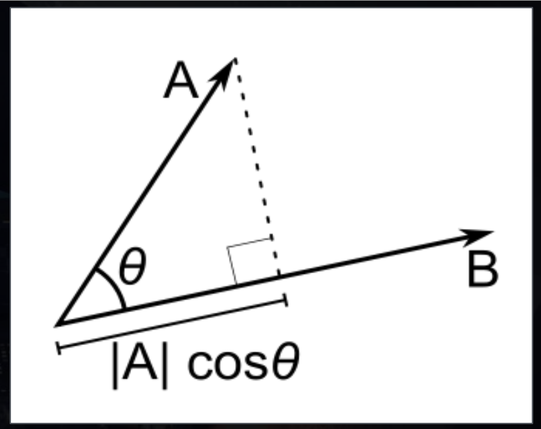
The Cosine Function



X-axis: Angle between `AIForwardIntention` & `TankForward`

Y-axis: `Throw` for `IntendMoveForward()`

The Vector Dot Product

Type	Input	Output	Interpretation & Notes	Diagram
Dot	2 x FVector	Float	**Projection of one vector onto the other, "parellness"** Maximum when vectors are parallel, and zero when they are perpendicular. Cos ***θ***	

$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \, \|\mathbf{B}\| \cos(\theta),$$

where θ is the angle between **A** and **B**.

In particular, if **A** and **B** are orthogonal, then the angle between them is 90° and

$$\mathbf{A} \cdot \mathbf{B} = 0.$$

At the other extreme, if they are codirectional, then the angle between them is 0° and

$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \, \|\mathbf{B}\|$$

Use `FVector::DotProduct()`

- Dot `AIForwardIntention` & `TankForward`
- Feed the result into `IntendMoveForward()`
- Share the resulting movement with the community.

Lecture Resources

01. Community Discussions (for intros, shares, discussions, tips etc)
02. Dot Product Scalar Projection on Wikipedia

4.12.4

`CrossProduct()` Vector Operator

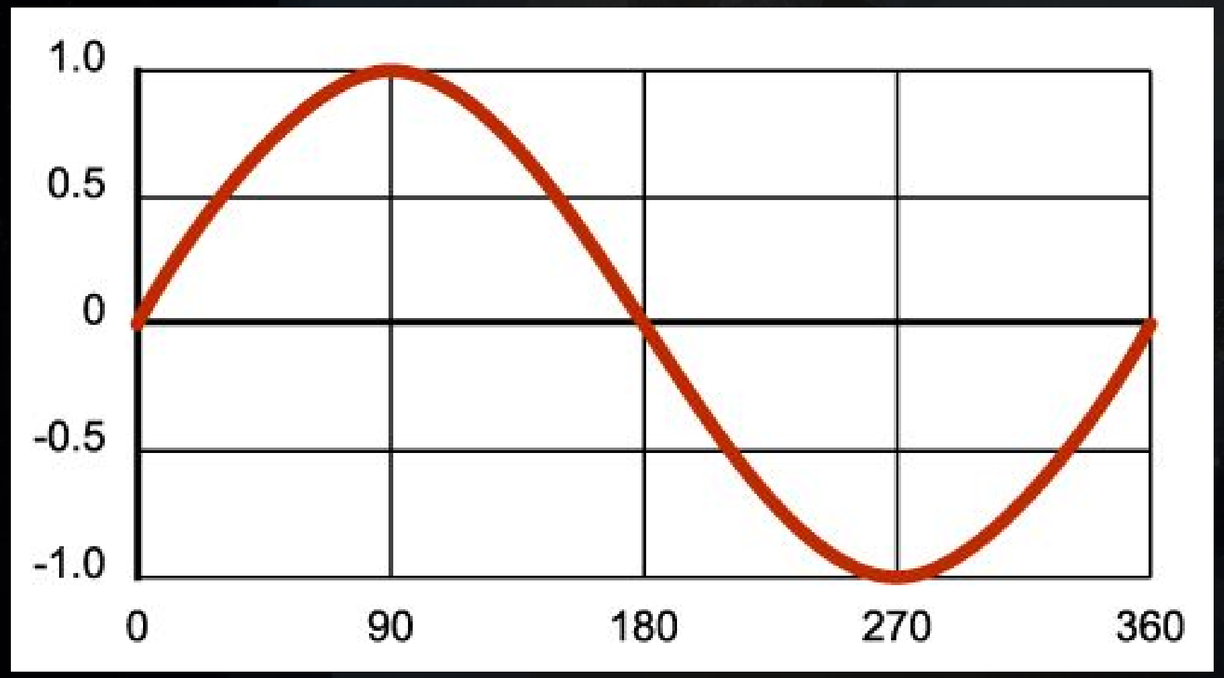
Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Focusing on controlling turning of AI
- If target in front or behind* don't rotate
- If target to side rotate at full speed
- This is the behaviour of a sin function
- Using `FVector::CrossProduct()`

** If behind once any rotation starts it will continue.*

The Sine Function



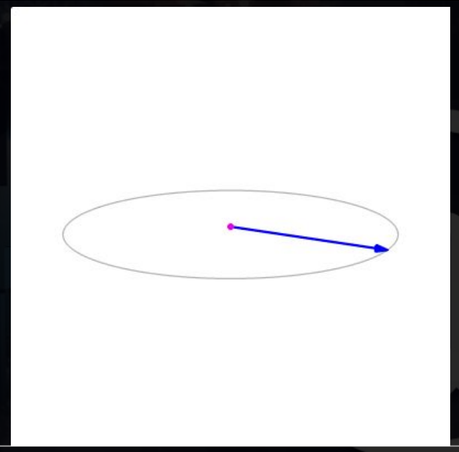
X-axis: Angle between `AIForwardIntention` & `TankForward`

Y-axis: `Throw` for `IntendTurnRight()`

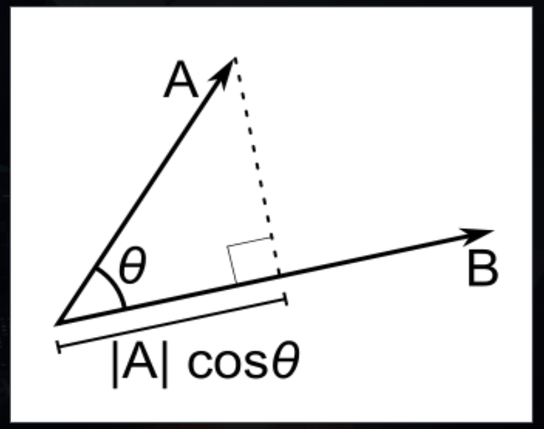
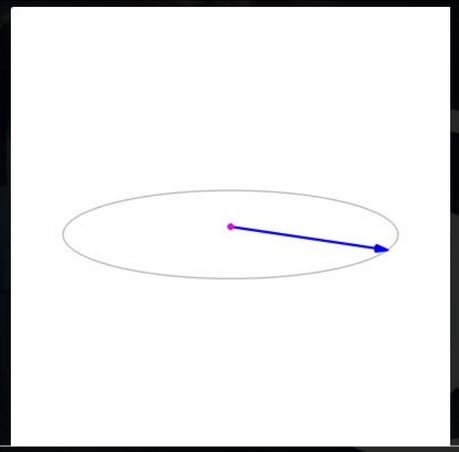
The Vector Cross Product

$$\mathbf{a} \times \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \sin\theta \, \mathbf{n}$$

where θ is the angle between **a** and **b** in the plane containing them (hence, it is between 0° and 180°), ‖**a**‖ and ‖**b**‖ are the magnitudes of vectors **a** and **b**, and **n** is a unit vector perpendicular to the plane

Type	Input	Output	Interpretation & Notes	Diagram
Cross	2 x FVector Order matters.	FVector	**"perpendicularity"** Maximum when vectors are perpendicular, zero when they are parallel. Always orthogonal to both. Sin ***θ***	

Cross & Dot Products Compared

Type	Input	Output	Interpretation & Notes	Diagram
Dot	2 x FVector	Float	**Projection of one vector onto the other, "parellness"** Maximum when vectors are parallel, and zero when they are perpendicular. Cos ***θ***	
Cross	2 x FVector Order matters.	FVector	**"perpendicularity"** Maximum when vectors are perpendicular, zero when they are parallel. Always orthogonal to both. Sin ***θ***	

Use `FVector::CrossProduct()`

- Cross `AIForwardIntention` & `TankForward`
- Find the Z component of the resulting vector
- Feed the result into `IntendTurnRight()`
- Share the resulting movement with the community.

Lecture Resources

01. Community Discussions (for intros, shares, discussions, tips etc)
02. Cross Product Animation on Wikipedia

4.12.4

Finalising Your Class Code

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Private, protected or public? Use the safest
- UPROPERY / UFUNCTION needed? Use “”
- `#include` and forward declarations required?

4.12.5

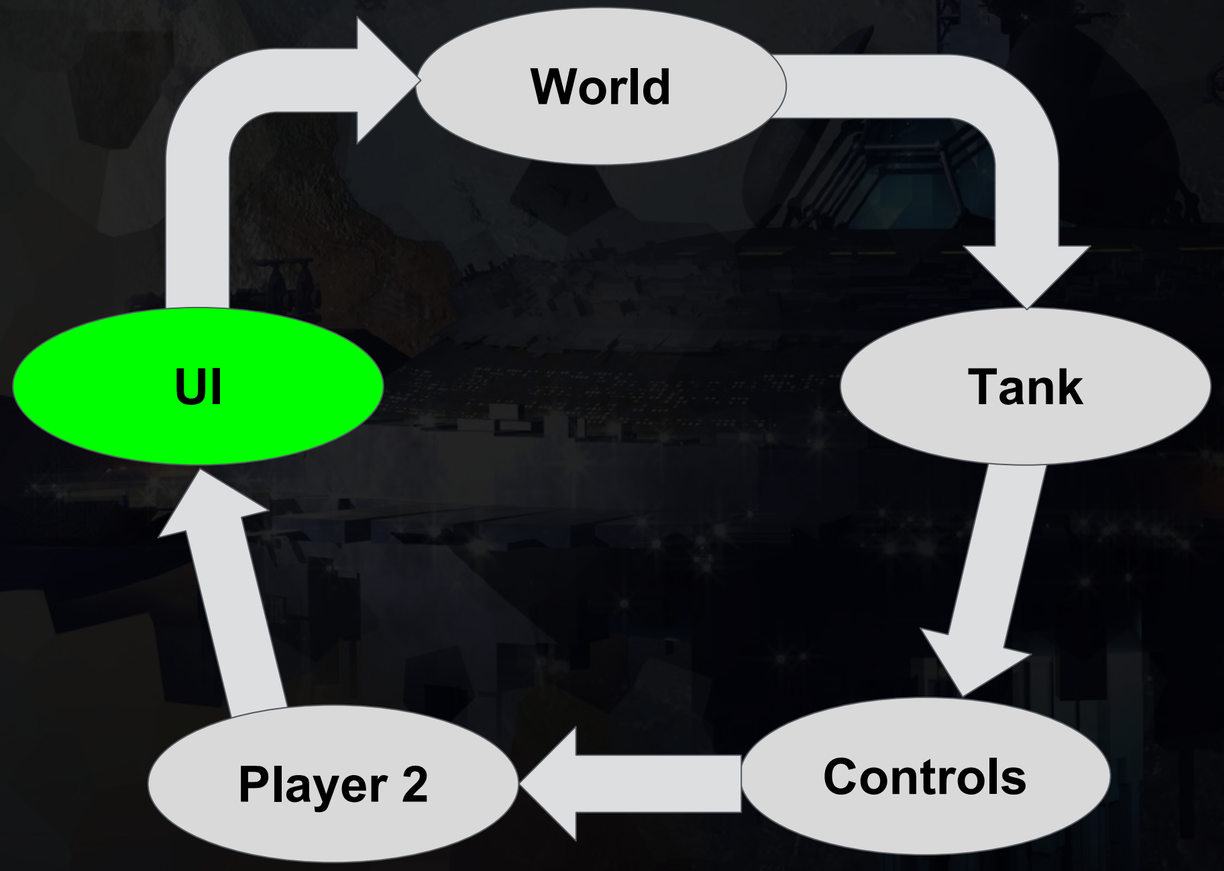
How to Use Blueprint Variables

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

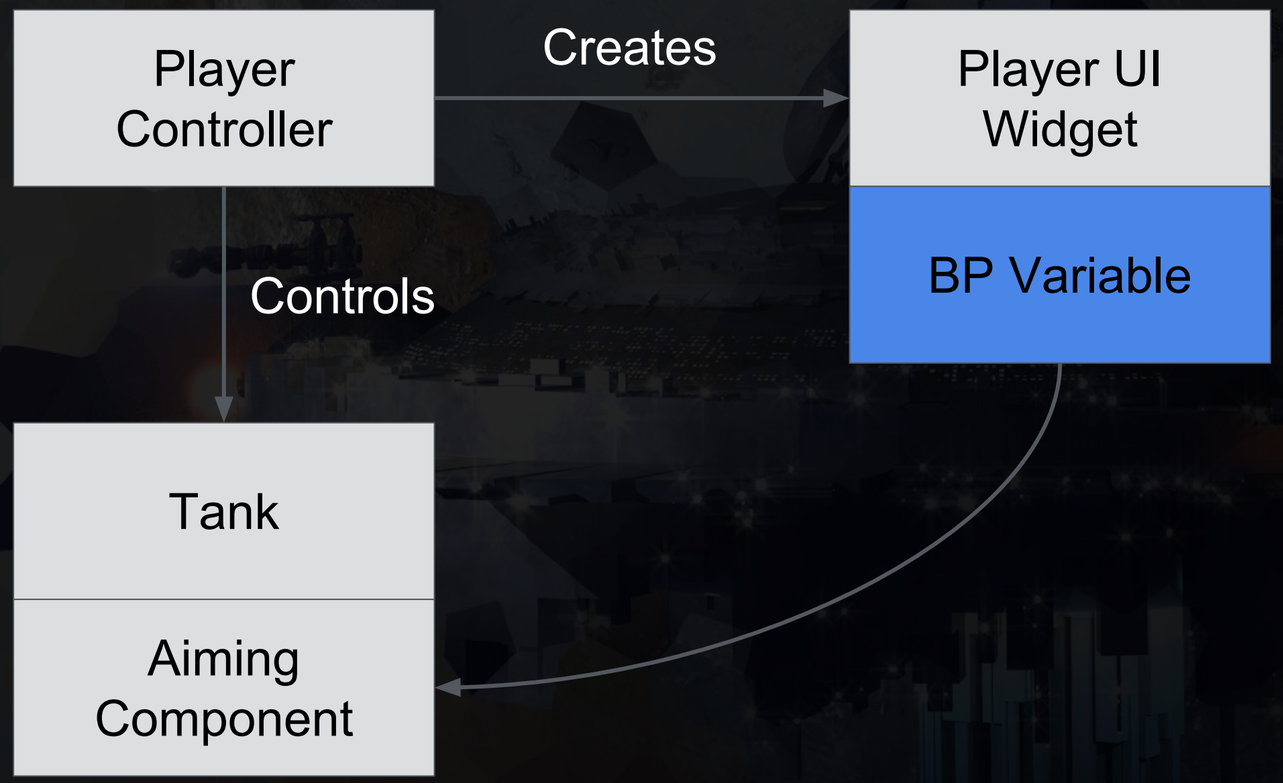
In This Video…

- Remember “what’s the least fun thing about this?”
- One thing is not knowing if you can fire
- How to change crosshair colour in blueprint…
- … according to the aiming component state
- States: Locked, Aiming, Reloading
- Referencing actor component from player UI.

Our Iterative Cycle



How Our Crosshair Gets Its Colour



Wire Up Get Aim Point Color...

- Wire the rest of this Blueprint
- Test that it works by changing the index and observing the crosshair colour change.

4.12.5

Using Enum(erations) in UE4

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- We met `enum class` around lecture 35
- In Unreal we must annotate with `UENUM()`
- We must specify the storage type (`uint8`)
- See Unreal's coding standards in Resources
- Remember we use enums to encode meaning.

Create `EFiringStatus`

- Use the following pattern...

```
UENUM()
enum class EThing : uint8 { Thing1, Thing2 };
```

- Specify Locked, Aiming and Reloading
- Create a private member & initialise to Reloading.

4.12.5

Refactoring our Aiming Component

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

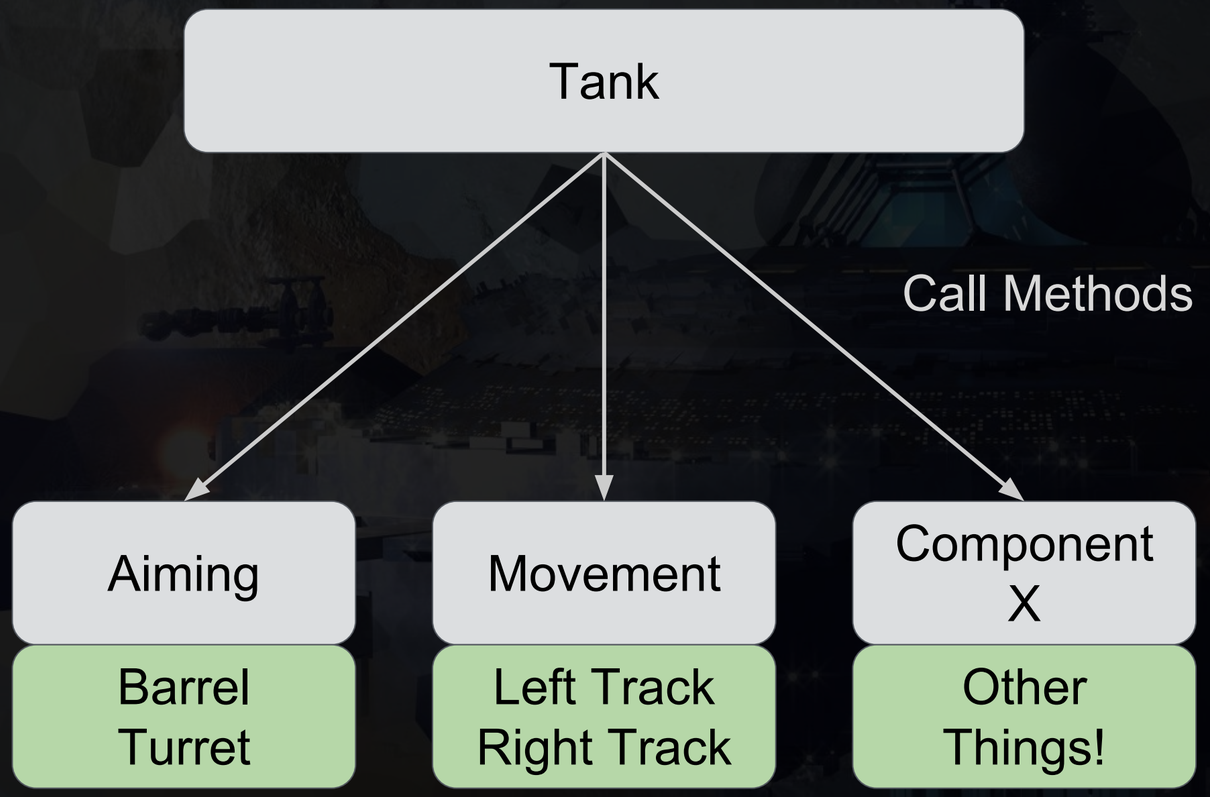
In This Video…

- Move away from `CreateDefaultSubObject()`
- Make aiming a `BlueprintSpawanableComponent`
- Get our code re-compiling as soon as possible
- Experience hard crash and add pointer protection
- Possibly get exasperated that we can’t find the suspected null-pointer causing the crash.

Refactor the Aiming Component

- Try and move the aiming component to the same architecture as the movement component, that is added in blueprint
- Don’t worry if you can’t get it working, but see if you can at least get it compiling and then commit.

Our Component Architecture



4.12.5

Attaching a Debugger to Unreal

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Hard crashes can be difficult to diagnose
- Attach your IDE’s debugger to the Unreal editor
- Use it to discover the source (often null pointer)
- We can also probe using Print in blueprint.

Add the Pointer Protection

- Protect the null pointer we have found
- Ensure the game still runs
- Don’t worry about setting the pointer yet

4.12.5

Constructor & Begin Play Timing

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Adding log entries to C++ and BP helps you to uncover the timing over events in the engine
- We’re doing this to discover exactly when Construct and Begin Play gets called in both C++ and Blueprint
- Note dropped actors are constructed in editor.

Convince Yourself with Logs

- Log Construct and Begin Play for the tank
- Do this both in C++ and Blueprint
- Use a unique prefix to filter logs
- See what order things happen in
- Also see the order from the Start Menu.

What Runs When...

Output Log

Filters donkey

```
LogTemp:Warning: DONKEY: Tank Tank_BP_569 C++ Construct
LogBlueprintUserMessages: [Tank_BP_569] DONKEY: Tank_BP Construct
LogTemp:Warning: DONKEY: Tank Tank_BP_C_12 C++ Construct
LogBlueprintUserMessages: [Tank_BP_C_12] DONKEY: Tank_BP Construct
LogBlueprintUserMessages: [Tank_BP_569] DONKEY: Tank_BP Begin Play
LogTemp:Warning: DONKEY: Tank Tank_BP_569 C++ Begin Play
LogBlueprintUserMessages: [Tank_BP_C_12] DONKEY: Tank_BP Begin Play
LogTemp:Warning: DONKEY: Tank Tank_BP_C_12 C++ Begin Play
```

Run from Main Menu

Tank 569 is the AI tank

```
LogTemp:Warning: DONKEY: Tank Tank_BP_569 C++ Construct
LogTemp:Warning: DONKEY: Tank Tank_BP_C_0 C++ Construct
LogBlueprintUserMessages: [Tank_BP_C_0] DONKEY: Tank_BP Construct
LogBlueprintUserMessages: [Tank_BP_569] DONKEY: Tank_BP Begin Play
LogTemp:Warning: DONKEY: Tank Tank_BP_569 C++ Begin Play
LogBlueprintUserMessages: [Tank_BP_C_0] DONKEY: Tank_BP Begin Play
LogTemp:Warning: DONKEY: Tank Tank_BP_C_0 C++ Begin Play
```

Run from BattleGround

4.12.5

Decoupling Your Architecture

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

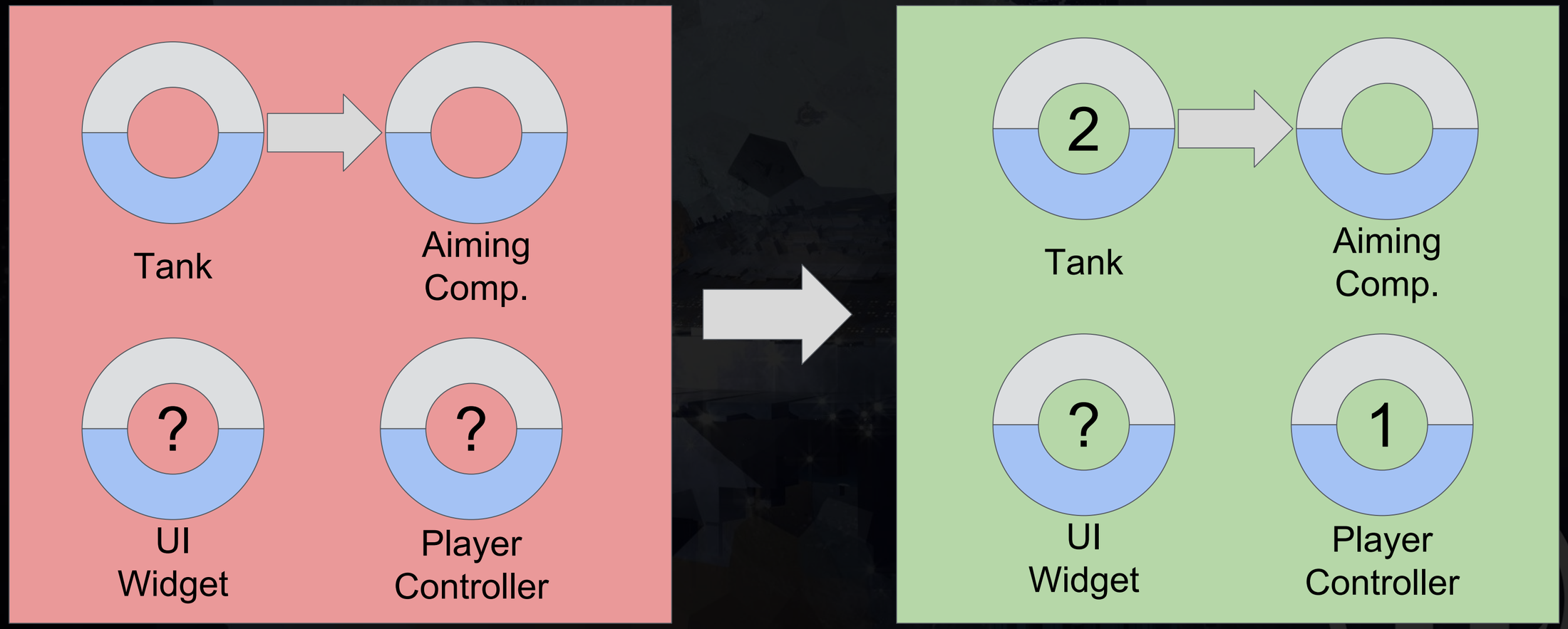
In This Video…

- We don’t have a Aiming Component reference
- It is hard to find a sensible time to set it
- Also we don’t ***need*** the reference on the tank
- We can Get Components by Class in Blueprint
- Mock-up our C++ code in Blueprint.

Constructors

Begin Play

Key Object Timing



Constructors

Begin Play

4.12.5

BlueprintImplementableEvent

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- We want to expose a C++ function to Blueprint
- We also want to pass a parameter (aiming ref.)
- Multicast delegates only work like this for actors
- We're using a component so we use…
- `UFUNCTION(BlueprintImplementableEvent)`
- You don't need to define the function!

Create UI Widget from Event

- Go to the TankPlayerController_BP
- Use the event to trigger UI creation
- Also use the passed reference to set the variable
- Test that the crosshair color now tracks the enum state set in the declaration.

4.12.5

Using the ensure Assertion

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- We use pointer protection to prevent hard crashes
- If we still get a crash we can attach the debugger
- The way we protect leads to things not working
- We may not notice this, which is dangerous
- So we use the `ensure` assertion macro.

Ben's Game Error Handling Criteria

- Execution should continue if possible
- We get a helpful warning with line #
- We can compile out the checks for production

```
if (!(LeftTrack && RightTrack)){ return; };
```

becomes...

```
if (!ensure(LeftTrack && RightTrack)){ return; };
```

Use `ensure` Everywhere

- Add ensure to all your pointer protection
- Find the null pointer that’s stopping aiming
- Celebrate!

4.12.5

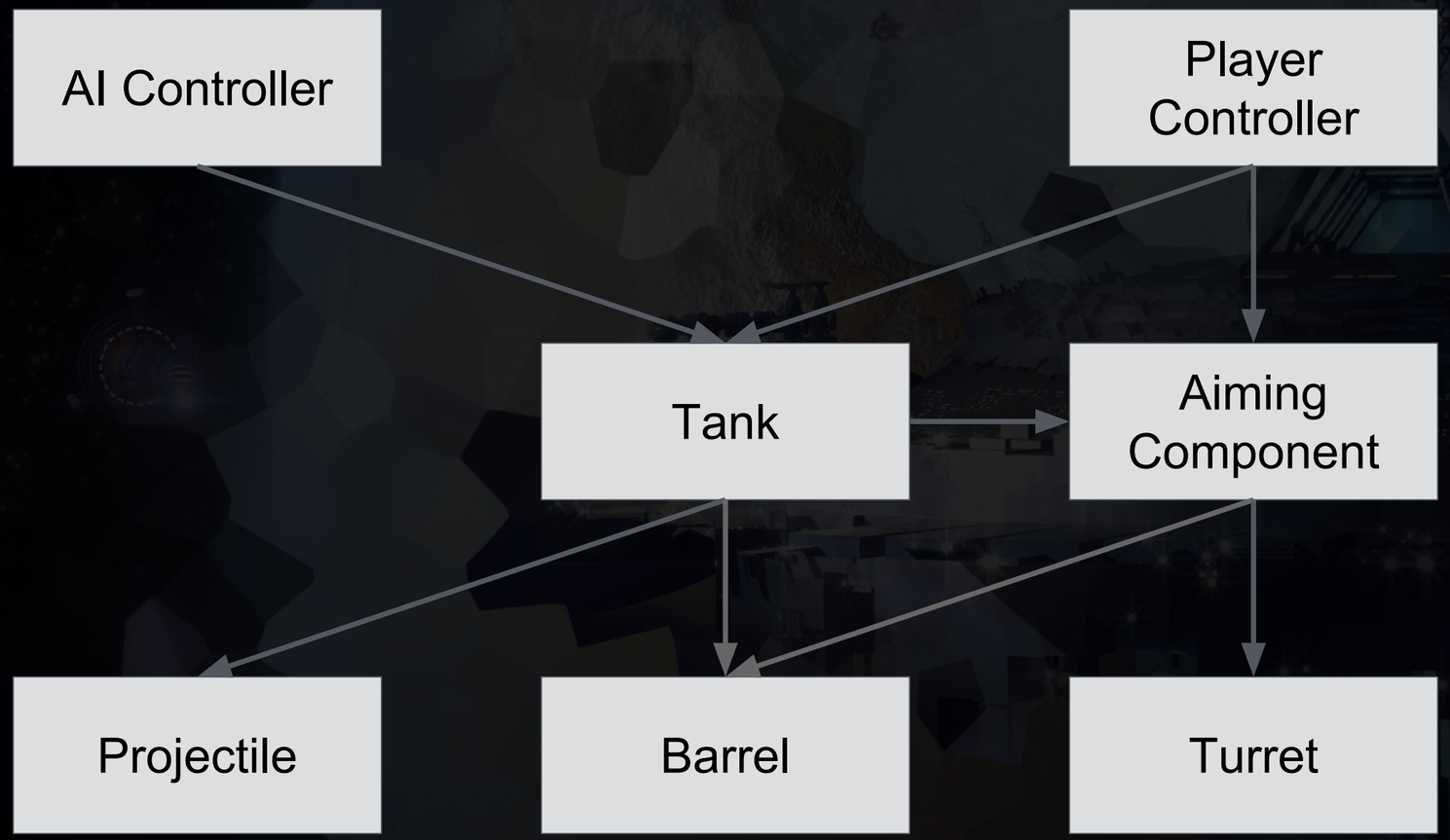
Dependency Mapping

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Code architecture can be hard to see
- Dependency mapping shakes-out the structure
- Go through your .cpp files and look at includes
- Map these as dependencies on a diagram
- If it looks like spaghetti, you need to refactor!

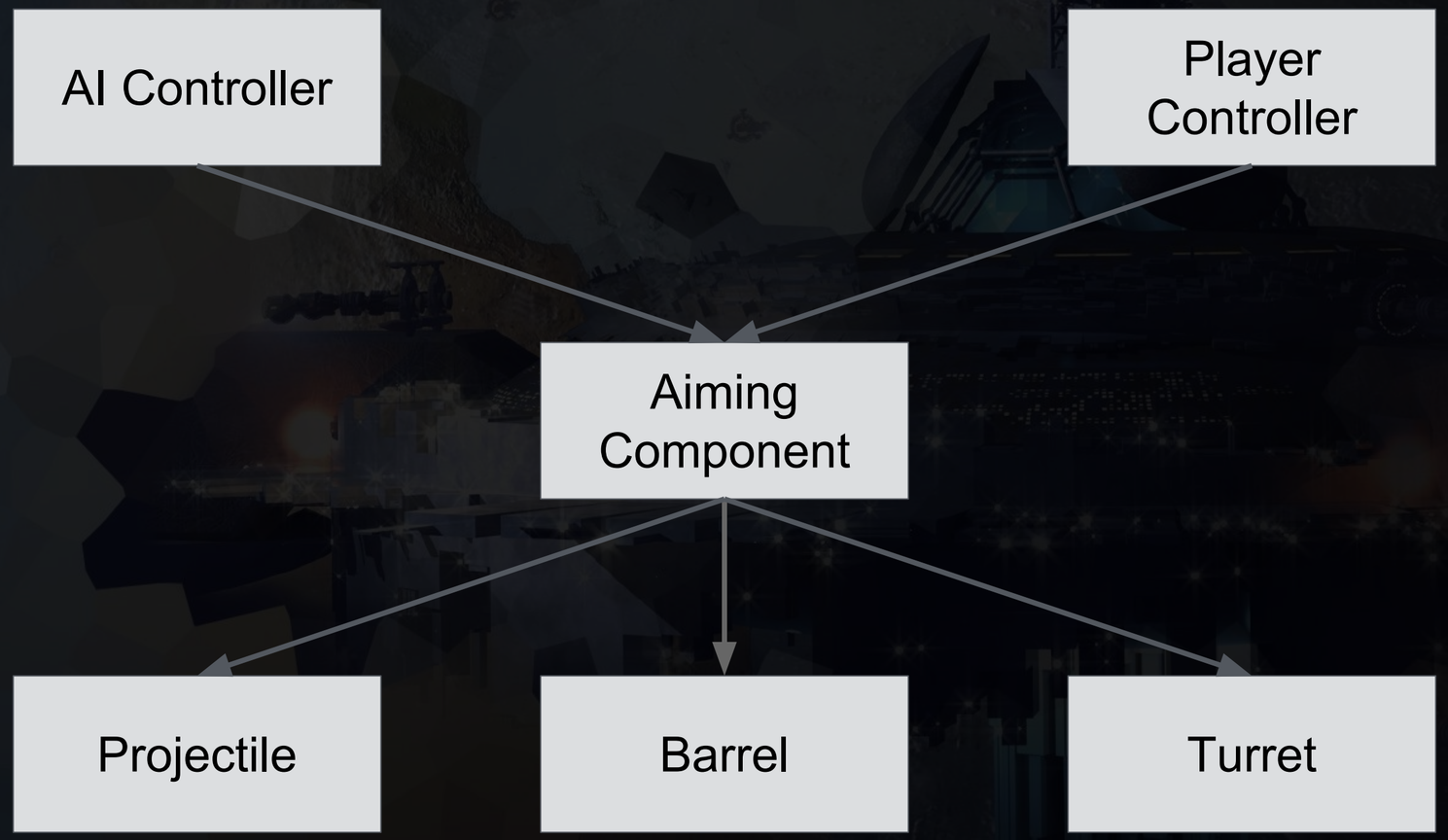
Our Current Aiming Architecture



Spaghetti!

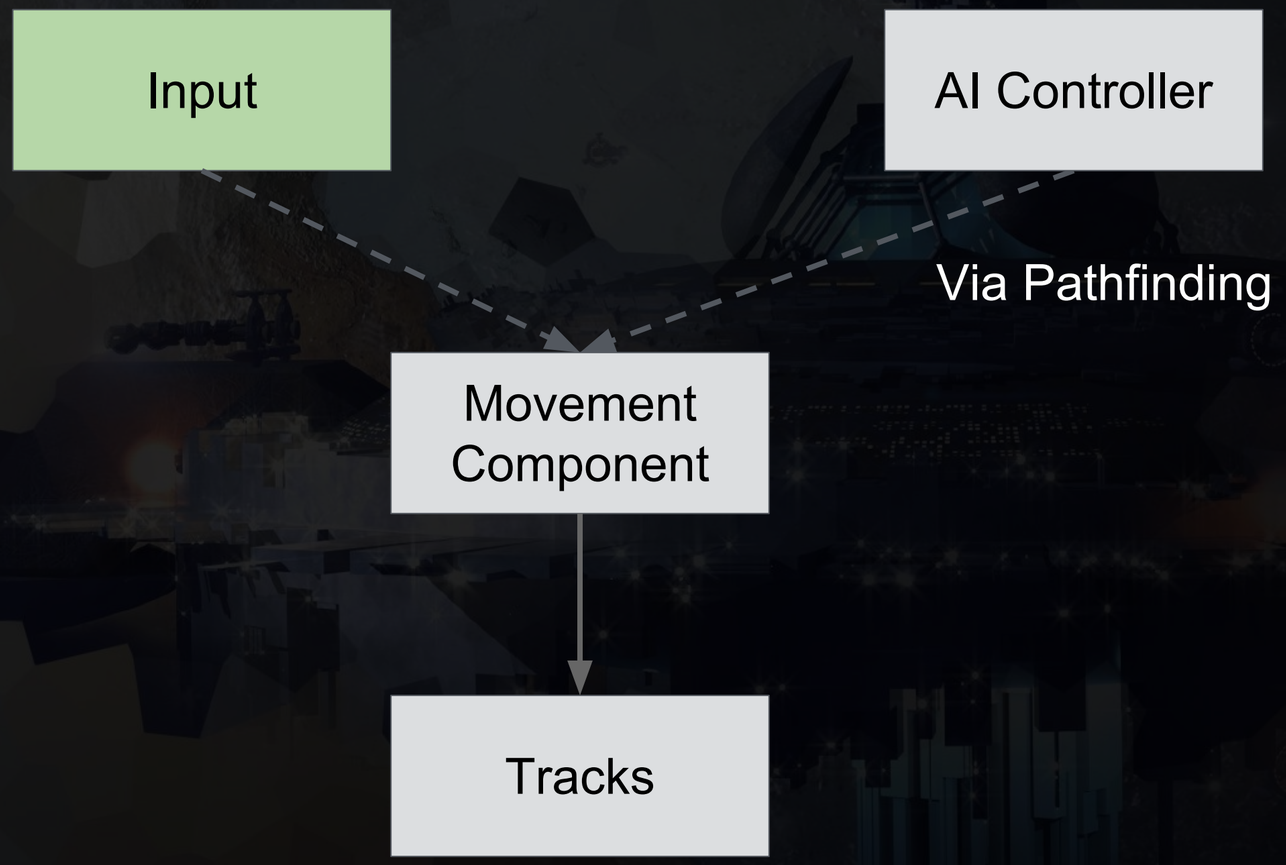
TankTrack and TankMovementComponent omitted for clarity

Our Desired Aiming Architecture



TankTrack and TankMovementComponent omitted for clarity

Our Desired Movement Architecture



TankTrack and TankMovementComponent omitted for clarity

4.12.5

Real World Skills

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Congratulations on getting this far
- We're not teaching sterile solutions here
- We're showing you how to recognise real issues
- … and how to tackle them sensibly
- It's not the easy path, but it is the valuable one.

4.12.5

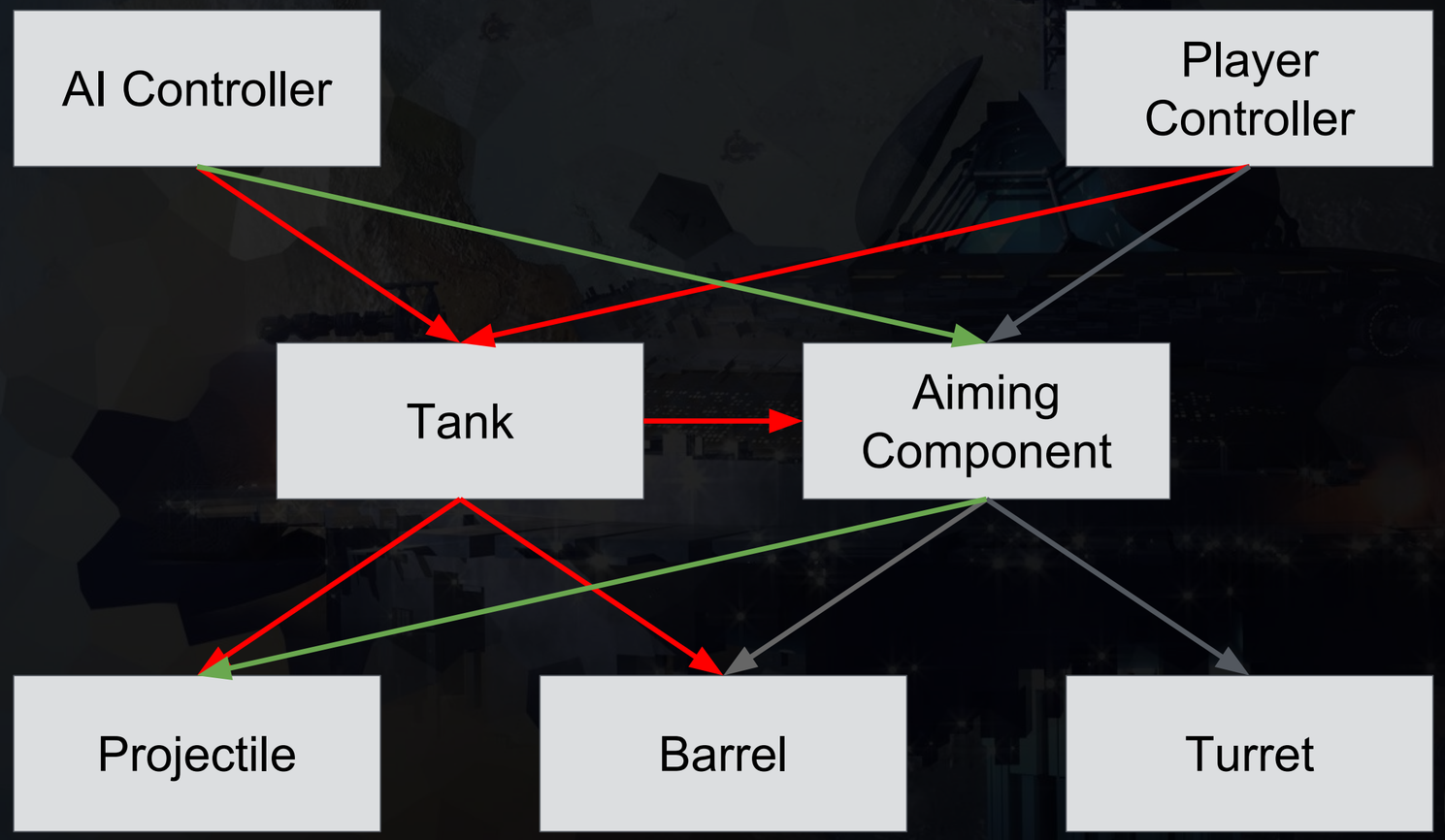
Starting From Green

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

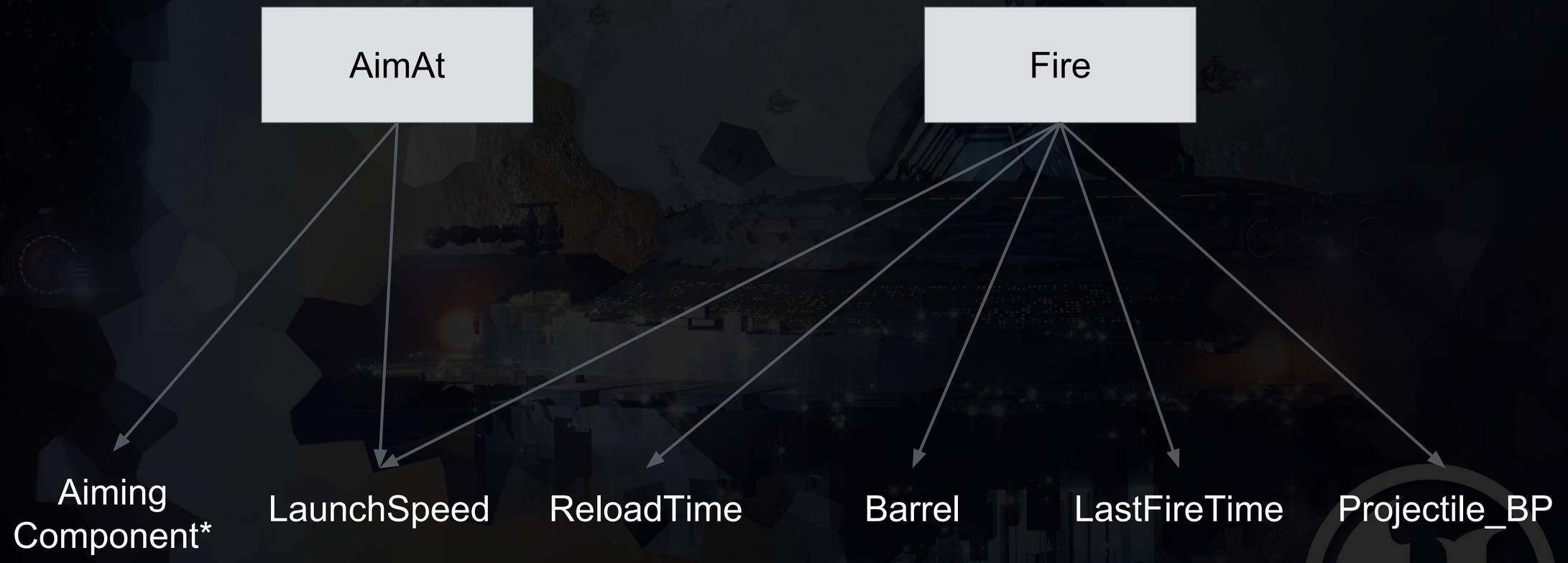
- You should probably only refactor working code
- Red means your code’s not working
- Green means it is, even if the code is messy
- We commit at green, then start refactoring.

What's Changing - Class Level



TankTrack and TankMovementComponent omitted for clarity

Aiming Methods in Tank Now...



TankTrack and TankMovementComponent omitted for clarity

Red -> Green -> Refactor

Red
Functionality in question isn’t working.

Green
Functionality is working, even if code is messy.

Refactor
Tidy the code, without changing functionality

TankTrack and TankMovementComponent omitted for clarity

Get Aiming Working

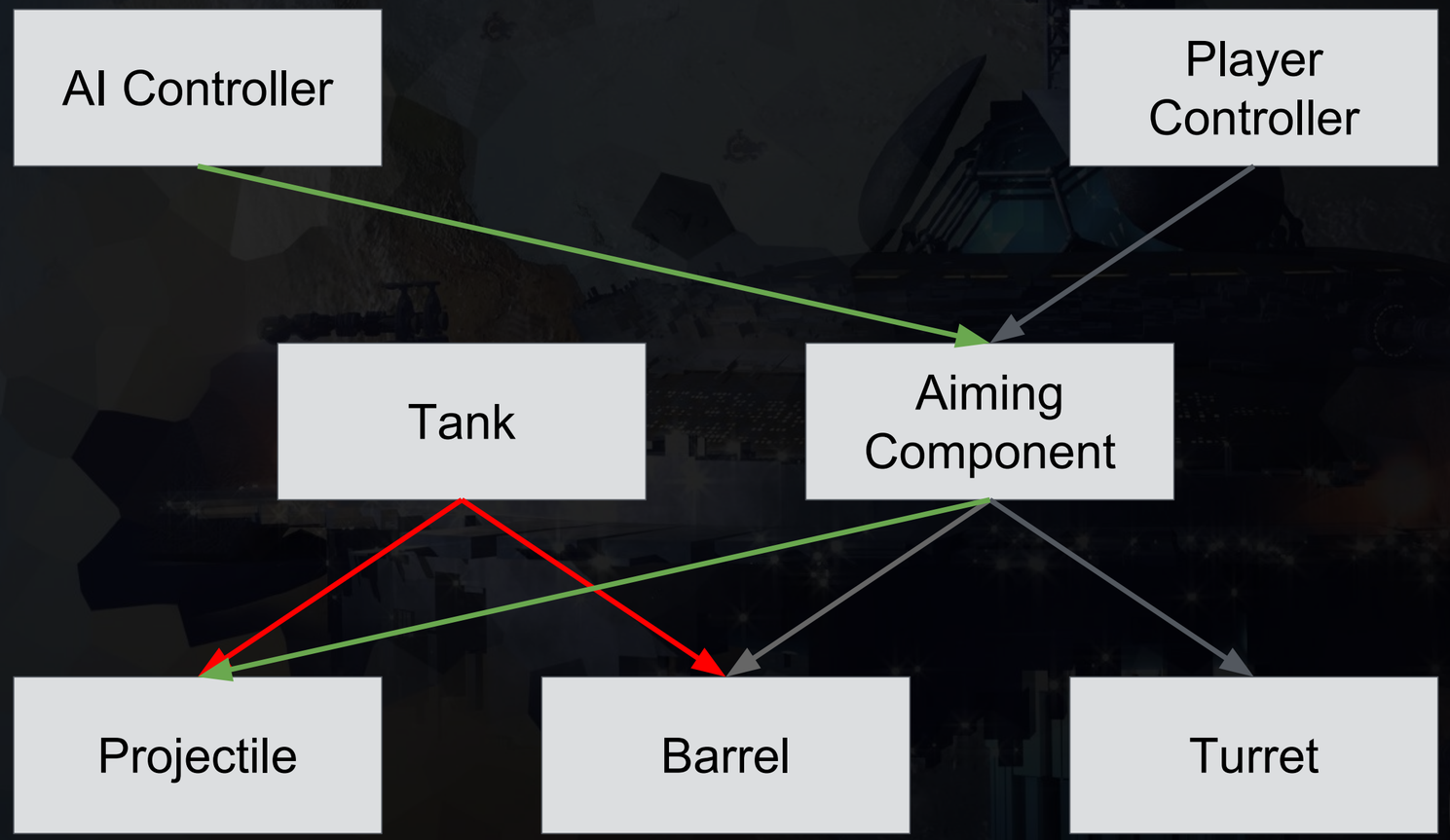
- If your aiming doesn’t work, get it working
- I introduced a bug in the aiming component
- It can be hard to track, use logs
- Follow the execution through.

4.12.5

Aiming Without the Tank

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

What's Changing - Class Level



TankTrack and TankMovementComponent omitted for clarity

In This Video…

- There is no need to cast the Pawn to a Tank
- Doing so creates a dependency we don’t want
- Remember a Tank is a Pawn
- We simplify our architecture here.

Aim Without the Tank!

- Move **just the aiming code** from tank
- Move forward declarations and includes
- Check all tanks now aim.

4.12.5

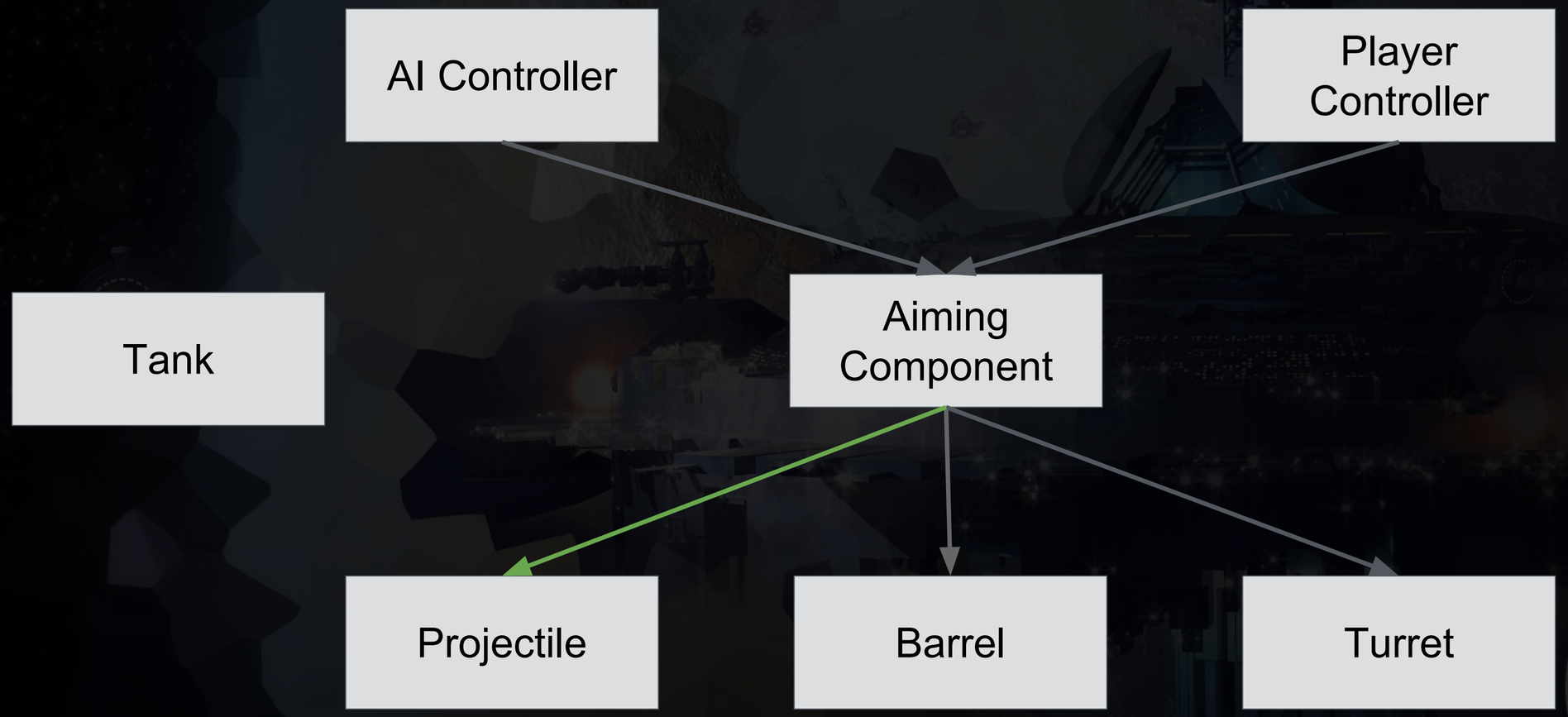
Finishing our Refactoring

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Removing our final dependencies
- If you override `BeginPlay()` in an actor you should call `Super::BeginPlay()`
- If you don't override it at all, there's no need to, your Blueprint Begin Play will still run.

What Needs To Change



TankTrack and TankMovementComponent omitted for clarity

Fire Without the Tank

- Move all remaining firing code out of tank
- The tank will end-up with no code useful code!
- Re-specify projectile in blueprint
- Re-bind the firing in blueprint
- Good luck!

4.12.5

Adding `TickComponent()` Back

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Actor Components use `TickComponent` not `Tick`
- You can find the signature in docs online
- Or by copying from the engine code
- Remember to use `override` at to check
- Remember to set the boolean in the constructor
- `GetWorld()->GetTimeSeconds()` alternative.

4.12.5

Are Two Floats Equal?

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- FVectors are just structs containing floats
- You must “define equal” when comparing floats
- The `FVector::Equals()` method allows this
- Specify a tolerance, see docs in resources.

Write `bool IsBarrelMoving()`

- Compare the barrel's current forward vector...
- ... to the `AimDirection` used in `AimAt()`
- Use the `FVector::Equals()` static method
- Test it works by seeing if it changes colour.
- Hint coming up next…
- Promote `AimDirection` to a member variable.

4.12.5

Programmatic Sideways Friction

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- We can apply a sideways correction force
- Remember **Force = Mass * Acceleration**
- … and Acceleration = Speed / Time
- So we calculate the force using the slippage speed,the frame time, and the tank mass
- A way to calculate is `FVector::DotProduct()`

Calculate `SlippageSpeed`

- The component of speed in the tank right direction
- If no slippage should be zero
- If sliding entirely sideways, should be speed
- In general use cos of the angle between velocity
- … and the sideways (right) vector
- Use `FVector::DotProduct()`

4.12.5

`OnComponentHit` Event in 4.12

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- We could use OnComponentHit in Blueprint
- But we're grown-ups so we're going to use C++
- Signature of `OnHit(...)` has changed in 4.12
- Remember you need to make it a `UFUNCTION`
- Details on next slide.

How to Register for `OnHit()` Events

1. Register delegate at begin play like this

```
OnComponentHit.AddDynamic(this, &UTankTrack::OnHit);
```

2. Use this signature for the delegate

```
OnHit(UPrimitiveComponent* HitComponent, AActor* OtherActor,
UPrimitiveComponent* OtherComponent, FVector NormalImpulse,
const FHitResult& Hit)
```

3. Make OnHit(...) a private UFUNCTION.

Register for `OnHit()`

- Use the instructions on the previous slide
- Ensure both tracks have the following set in BP…
- **Collision > Simulation Generates Hit Events**
- Test it works with a log entry
- Celebrate being one of the first since 4.12!

4.12.5

Avoiding Boolean Flags

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Boolean flags usually make answers old
- Try and think of a way of avoiding them
- Revise the use of `FMath::Clamp()`

4.12.5

Improving Tank Aiming

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Use a literal glass ceiling to help with testing!
- Sometimes the barrel takes the long route
- A simple `if()` statement can help here
- Find and fix another bug in the code
- You can use `%i` formatter to log booleans.

Make Turret Take Short Route

- Use a simple `if()` statement
- Rotate so that you always take the short route
- Test with player and AI tanks.

4.12.5

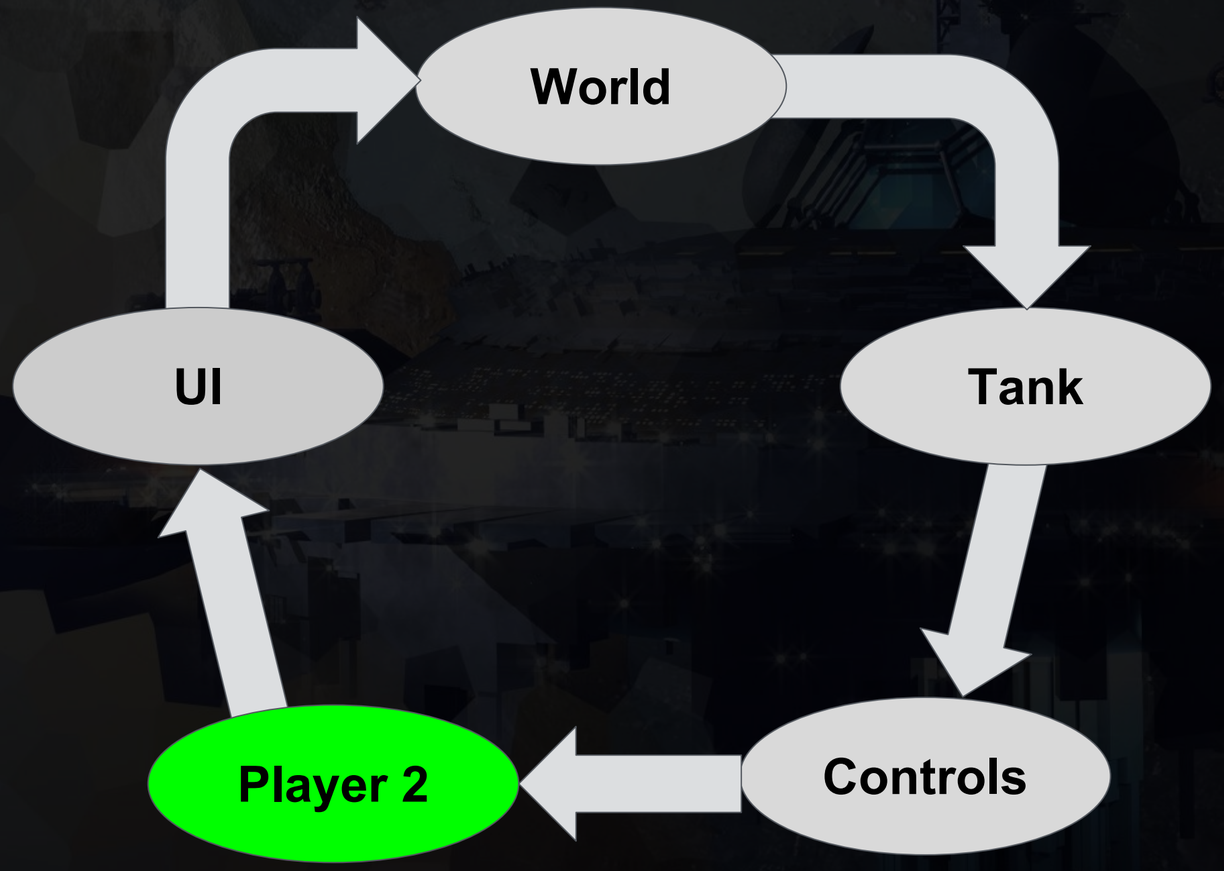
Tweaking Tank AI

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Expose the Acceptance Radius to blueprint
- Tweak that value as `EditAnywhere`
- Change back to `EditDefaultsOnly` once found
- Prevent AI tanks firing until aiming is locked.

Our Iterative Cycle



Choose Your Acceptance Radius

- Make a blueprint of the TankAIController class
- Expose `AcceptanceRadius` as `EditAnywhere`
- … at least for now to help you tweak the values
- Code your default Acceptance Radius
- Place your AI tanks in the world
- Hint: Remember to refresh navmesh if required.

4.12.5

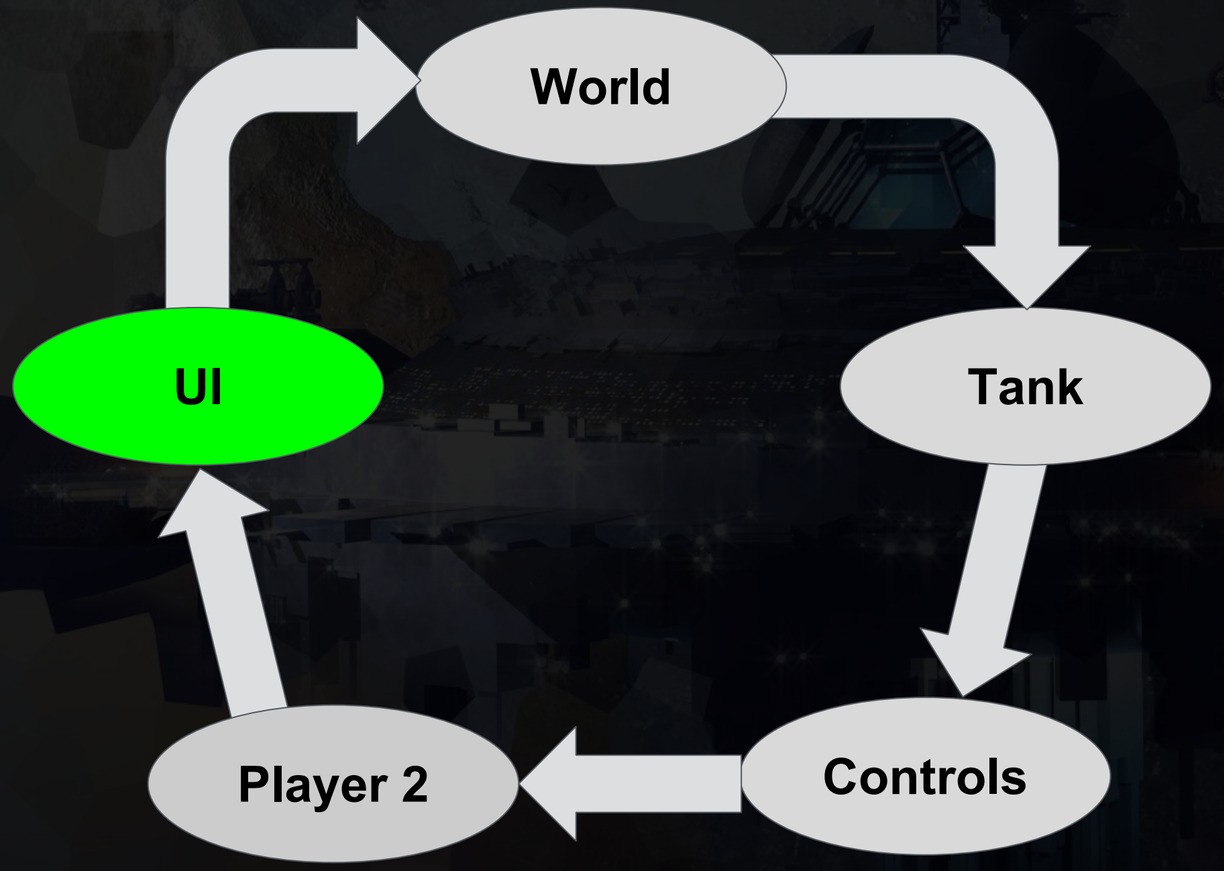
Making an Ammo Display

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Add a 4th enum state for out of ammo
- Work around bug of blueprint select nodes not updating when we add new enum values in C++
- Add a display for rounds remaining
- Bind the UI to a `GetRounds()` method.

Our Iterative Cycle



Make an Ammo Display

- Get an ammo display working
- Should count down each time you fire
- Bonus: make crosshair grey when out of ammo
- Tip: restart editor after adding new enum state.

4.12.5

Making an AutoMortar

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- The tank components were built for the tank
- It turns-out we can re-use movement and aiming
- This is the benefit of re-usable components
- We'll create a self-aiming mortar tower.

Assemble the AutoMortar

- Look at how the Tank_BP is constructed
- Put the mortar together in a similar way
- Use the Tank Aiming Component
- Associate with the Tank AI Controller
- We'll rename these classes once it's working
- Bonus: get the AutoMortar actually working.

4.12.5

Using the Reference Viewer

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Currently we .gitignore the Starter Content
- Therefore we can’t track changes
- We want a consistent starting point for particles
- So we’re going to delete Starter Content
- Lots depends on it so we use a special tool
- That special tool is the reference viewer.

Remove your Starter Content

- Backup the folder as a .zip outside project
- Try deleting Starter Content in Content Browser
- Move all referenced assets out or delete objects
- Delete the Starter Content folder
- Re-open the editor and test everything works
- Share you tidy folder structure in the discussions.

4.12.5

Preparing for Particles

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- We will compose our projectile in C++
- Use `SetRootComponent()`
- Use `AttachTo(RootComponent)`
- You can set default properties in C++
- Use `UPROPERTY(VisibleAnywhere)`

Setup the Projectile Components

- Create default sub-objects in C++ for…
- `UStaticMeshComponent` called `CollisionMesh`
- `UParticleSystemComponent` called `LaunchBlast`
- Make them `UPROPERTY(VisibleAnywhere)`
- Check they appear on the Blueprint.

4.12.5

Introducing Particle Systems

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Setup a Starter Content project
- Use it to migrate assets to Battle Tank
- Explore particle systems
- Use world space for smoke trails
- Create and share your smoke trail.

Setup & Share Your Smoke

- Tweak the parameters to your taste
- Get creative with color, size, etc
- Don’t worry about it disappearing from side view
- Share with the community (link in Resources).

4.12.5

Particle Bounding Boxes

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

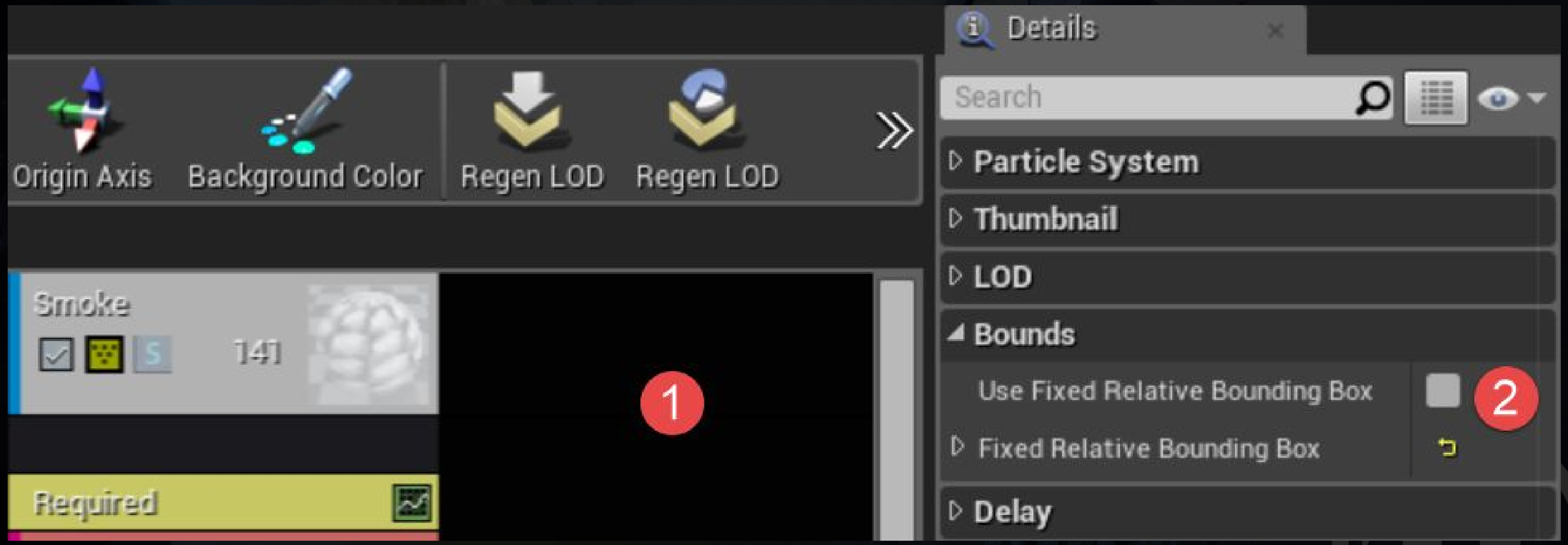
In This Video…

- Our smoke disappeared when viewed from side
- This is due to the fixed particle bounding boxes
- We can fix it by making the boxes dynamic
- BUT we need to remove GPU rendered particles
- … and test the performance hit is acceptable.

Test Your Performance Impact

- Measure your FPS with 20 smoke trails visible
- Turn on dynamic bounding boxes
- Measure again, see if there’s a change
- Share your results with the community.

How to Set Dynamic Bounding Boxes



4.12.5

Using `FAttachmentTransformRules`

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Use Message Log to see warnings
- AttachTo() has become `AttachToComponent()`
- Now must provide `FAttachmentTransformRules`
- We'll use `KeepRelativeTransform` for now
- Write code to de-active launch blast and
- Activate impact blast on impact.

Code Projectile Impact

- Register a component `OnHit()` event delegate
- Use `LaunchBlast->Deactivate()` to turn off
- Add an `ImpactBlast` particle effect to projectile
- Activate this on impact
- Share your explosion.

4.12.5

Radial Forces & Caching

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- If you don't `AttachToComponent()` then…
- It will look like you're attached but…
- The transform may be broken and…
- You'll get really weird effects and…
- Unreal may cache the issue
- So, always `AttachToComponent()` :-)

Find the Bug

- Why is this behaviour happening?
- Try inspecting the projectile at run time
- Look carefully at the Explosion Force component
- Write a line of code to solve the problem
- Share your solution in the discussions.

4.12.5

Using `GetTimerManager()`

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Currently we don’t destroy our projectiles
- This will cause slow-down and memory leakage
- You won’t pass console testing with a leak
- Tidy up after ourselves
- Discuss projectile schemes
- Destroy our projectiles with a timer.

Projectile Destruction Schemes

Sheme	Pros	Cons
Projectile destroy timer	Simple. Projectile self-responsible.	Not very performant with 100s of projectiles.
Spawn explosion effect	More physical, projectile gone, explosion remains.	More complex. Can cause slow-down just when you don't want it.
Projectile pool	Probably most performant when there's a natural limit to number of projectiles.	Most complex, consider a "ring buffer" or similar.

Make Projectiles Self-Destroy

- `GetWorld()->GetTimerManager().SetTimer()`
- Set a Blueprint editable `DestroyDelay`
- Write a simple delegate method
- Use `Destroy()` to destroy the projectile
- Test it works, and comment on approach.

4.12.5

Using `TakeDamage()` on Actors

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Unreal has an actor damage system
- We'll apply radial damage from the projectile
- Then the `AActor::TakeDamage()` method will be called on the tank (and all other actors in radius)
- We'll then finish our damage system off
- Solve the `int` or `float` damage question.

Implement a Damage System

- Override the Tank's virtual `TakeDamage` method
- Clamp `DamageToApply` to `CurrentHealth`
- Log `DamageAmount` and `DamageToApply`
- Carry on watching the video, or finish the system.

4.12.5

BlueprintPure & Health Bars

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Add a UI Widget component to our tank
- Make a very simple health progress bar
- Wire the bar to the tank.

Set the Tank BP Variable

- Make a Blueprint Variable on the Health Bar
- Set this from the Tank Event Graph

4.12.5

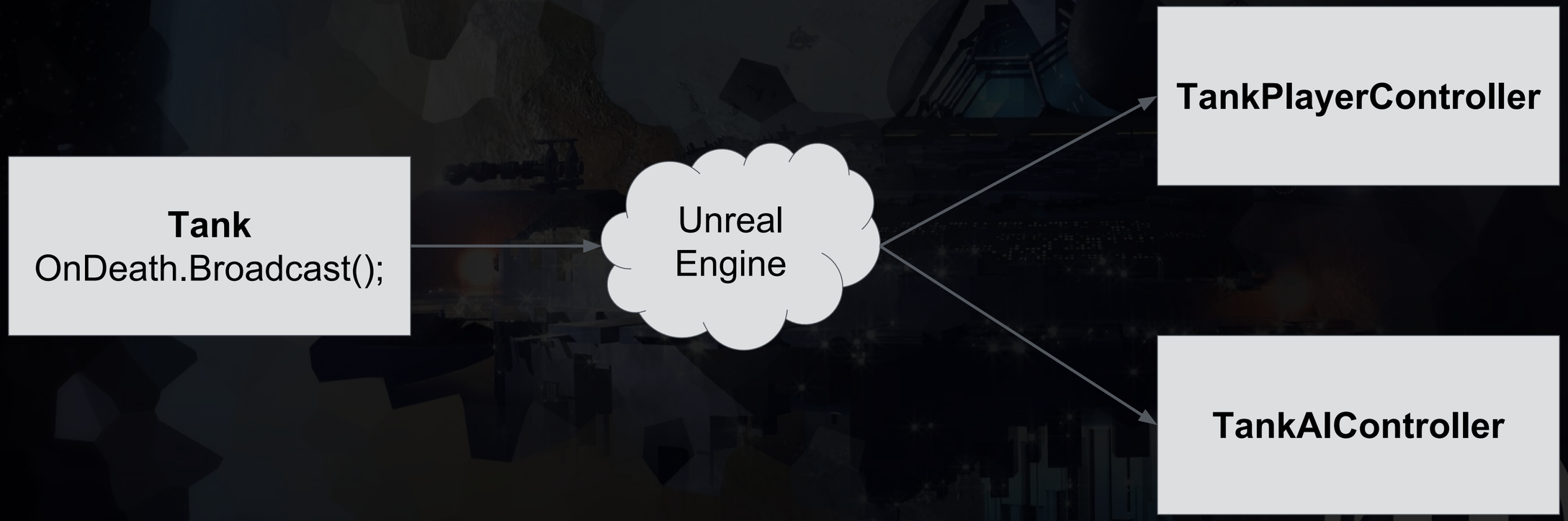
The Observer Pattern

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- We want the tank to broadcast it's death
- Tank doesn't need to know who is listening
- Player and AI controllers listen, and de-possess
- Other systems (e.g. audio or scoring) could listen
- `DECLARE_DYNAMIC_MULTICAST_DELEGATE`
- Binding and broadcasting with delegates.

How Our Delegates Work



Steps for Setting-Up DMCDs

1. Create type: `DECLARE_DYNAMIC_MULTICAST_DELEGATE(FSubjectName);`
2. Declare: `FSubjectName OnSomethingHappened;`
3. Broadcast: `OnSomethingHappened.BroadCast();`

---

4. Declare delegate on listener(s):

```
UFUNCTION()  // Must be a UFUNCTION to get called
void DelegateMethod();  // e.g. void OnTankDeath();
```

5. To register: `BroadcastingInstance-> OnSomethingHappened.`

```
AddUniqueDynamic(this, &AListenerClass::DelegateMethod);
```

Setup Delegate System

- Use the crib-sheet on the previous slide
- Register the TankPlayerController and the
- TankAIController to receive OnDeath
- From their possessed tank
- Log to prove they’re getting the message.

4.12.5

Finishing Off - Part 1

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- We're nearing the end of the section
- You have several challenges over to try
- These include various fixes and improvements...
- Use `StartSpectatingOnly()` in Player Controller
- `DetachFromControllerPendingDestroy()` in AI
- Fixing a bug with our starting health.

Pick & Mix Challenge

1. Call `StartSpectatingOnly()` in Player Controller
2. `DetachFromControllerPendingDestroy()` in AI
3. Fix the starting health bug
4. Use the noise landscape function to undulate
5. Make the AutoMortars damageable
6. Check code against Unreal's coding standards.

4.12.5

Finishing Off - Part 2

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- You can use the noise function on landscapes
- Gameobjects are automatically destroyed when they travel a long way from the play area
- Reviewing Unreal's coding standards.

Pick & Mix Challenge

1. ~~Call `StartSpectatingOnly()` in Player Controller~~
2. ~~`DetachFromControllerPendingDestroy()` in AI~~
3. ~~Fix the starting health bug~~
4. Use the noise landscape function to undulate
5. Make the AutoMortars damageable
6. Check code against Unreal's coding standards.

Section Wrap-Up

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

In this section we learnt a tonne including…

- Basic terrain landscaping
- Using and modifying the AI pathfinding system
- A deep-dive into control systems
- User Interface for the first time
- A whole tonne of C++ and architecture.

Improve Building Escape

- Put the project under version control
- Add a start menu
- Create at least one new puzzle
- Add some landscape around the room(s)
- Consider a projectile-based puzzle
- Improve the overall architecture.

4.12.5

Bonus - Switching Cameras

Twitter **@GameDevTV** :: Web **community.GameDev.tv**

In This Video…

- Our player controller line traces to aim
- This can hit the UI in some circumstances
- Change our line trace channel to `ECC::Camera`
- Add a 1st person camera
- Use the Toggle Visibility Blueprint node
- Bind input and enjoy simple camera swapping.

Write the Blueprint

- Search for a “toggle” node
- See if you can make it work.

INTENTIONALLY BLANK

… but now it isn’t - weird.

This is just to help mark the current end of the slides.

I’m adding content most weekdays