UnrealCourse.com Section 3 Slides - Building Escape

<< To Section 2 To Section 4 >>

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

v.4.10

Section Introduction

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

In This Video…

- Welcome to our first Unreal editor section
- You'll learn simple level building
- We'll be using meshes and materials
- C++ events accessed from Blueprint
- Calling C++ code from Blueprint
- And much more.

v.4.10

Building Escape Overview

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

In This Video…

- An overview of the finished project.

v.4.10

Game Design Document (GDD)

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

Concept

- The core concept is simple: escape the room
- You awaken in a locked room, unable to escape
- Use environmental clues such as light and sound to determine what to do next
- Trigger pressure plates and solve puzzles to progress towards the exit.

Rules

- No lose condition, apart from the feeling you’re going to die in this room if you don’t get out!
- Anything that you can do, you are allowed to do
- You win by finally exiting the room.

Requirements

- Unreal's provided Starter Content pack
- C++ code and Blueprint to encode behaviour
- Various sound effects to enhance atmosphere
- Sketches for layout of room(s)
- Sketches for how puzzles work.

Sketch Your Room(s)

- Sketch out one large room, or a few smaller ones
- Annotate where the puzzles will be
- You can change your mind later
- Create a more detailed sketch of one puzzle
- Share your sketches in the discussions
- Evernote can be great for storing these things.

Possible Future Ideas (The NO List)

- This is ready to capture crazy ideas as they come!

v.4.10

Version Control 101

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

An Overview of Source Control

- The what and why of Version Control Systems
- Choosing your Version Control System (VCS)
- What files to include / exclude
- Commit = save a local snapshot
- Reset = roll-back to a previous state
- Branch, Push and Large File Support later.

Popular Version Control Systems

- Git
- Mercurial
- Perforce
- Subversion / TortoiseSVN
- Alienbrain (for art but of order $10,000)

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

About SourceTree

- Free software by Atlassian
- Visual front-end for Git or Mercurial
- Mac and PC but Mac version is a little ahead
- Good when learning as easy to visualise.

Install Your VCS

- Pick a VCS for yourself
- We’ll be using Git with SourceTree as a front-end
- Install and register it
- Have a quicky play / experiment
- Carry on watching the videos.

v.4.10

Ignoring Unreal Derived Files

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

In This Video…

- Derived files can be easily rebuilt
- Other files (code, assets, level layout etc) can’t
- Ignore most derived files for version control
- Which folders to ignore in version control
- Our starting .gitignore file for Unreal.

Derived Folders In Unreal

- Binaries
- Build
- DerivedDataCache
- Intermediate
- Saved

https://docs.unrealengine.com/latest/INT/Engine/Basics/DirectoryStructure

v.4.10

Your First `.gitignore` for Unreal

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

In This Video...

- Understand Unreal creates VS projects for us
- How to re-generate VS project files
- Writing our first `.gitignore` file
- “Committing” our project for the first time.

Make Your First Commit

- Catch-up with what I did on this video
- Get your .gitignore file working*
- Add (stage) all your files and commit.
- Celebrate entering this bewildering new world!

** mine is attached to the first lecture of the section.*

Getting to Know Unreal’s Editor

@UnrealCourse :: www.UnrealCourse.com

In This Video…

- Why changes to the starter scene aren’t tracked
- Arranging a simple set of windows
- Moving around in the 3D Viewport
- Setting our start map, and committing.

Explore the 3D Viewport

- Use the little ? crib-sheet
- Explore until you're comfortable
- Add a few Props from the Starter Content.

A Pointers Primer

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

In This Video…

- You’re about to meet pointers for the first time
- The clue is when you see a * next to a type
- Pointers are simply memory addresses
- You have to “follow” the pointer to the object
- Benefit: saves you from moving things in memory
- Disadvantage: you can lose control of data.

Pointer Syntax

```
FActorComponentTickFunction* ThisTickFunction
FActorComponentTickFunction * ThisTickFunction
FActorComponentTickFunction *ThisTickFunction
```

- All three statements are equivalent, we use 1st
- In all cases ThisTickFunction is a pointer
- In all cases the type of the object pointed to is `FActorComponentTickFunction`

The `->` Accessor Operator

- Imagine we have `AActor* SomeActor;`
- The `AActor` class has a method `GetName()`
- `*SomeActor` “de-references” the pointer
- You could write `(*SomeActor).GetName();`
- But you can follow and access in one using `->`
- We access name with `SomeActor->GetName()`

Read More About Pointers

- http://www.cplusplus.com/doc/tutorial/pointers
- Share your understanding in the discussions
- Keep an eye out for a pointer in `PositionReport.cpp` in the next video.

v.4.10

Unreal's Class System

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

In This Video...

- Introducing the idea of inheritance
- Unreal’s scarily powerful class system
- Exploring using the Class Viewer*
- Inheritance for “is a” relationships
- Components for “has a” relationships.

https://docs.unrealengine.com/latest/INT/Engine/UI/ClassViewer/index.html

Inheritance for "is a" Relationships

- e.g. `Character` "is a" `Pawn`, `Pawn` "is an" `Actor`
- c.f. `Dog` "is a" `Mammal`, `Mammal` "is an" `Animal`
- Unreal makes extensive use of inheritance
- Is a powerful tool if used properly
- Can be inflexible and hard to re-factor.

Components for “has a”

- The chair & the rock “has a” PositionReporter
- Objects become rich through many components
- Can be flexible if used properly.

Take a Look at the Generated Code

- Take a brief look at `UPositionReporter.cpp`
- Take a shorter look at `UPositionReporter.h`
- Share what you recognise in discussions
- Share what you don’t recognise
- We’ll explore the files in the next video.

v.4.10

Runtime Messages for Feedback

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

In This Video...

- Using UE_LOG to print to the Output Console
- Printing to the game screen

For more information read...

https://wiki.unrealengine.com/Logs,_Printing_Messages_To_Yourself_During_Runtime#Related_Tutorial

Add Component to 2nd Object

- Add our new component to a 2nd game object
- If it works you’ll get a 2nd log
- We’ll see next how to read the object name.

v.4.10

Accessing Object Names

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

In This Video...

- Use `GetOwner()` to find the component's owner
- `AActor*` is a pointer to an actor, a new concept
- Use `->` to access methods through pointers
- Use `GetName()` to find the object's name
- Use `%s` as a format operator for strings
- Use `*` to "dereference" pointers.

v.4.10

Include What You Use For 4.17+

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

v.4.10

Getting Transforms in C++

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

In This Video…

- Introducing `FVector`
- Mixing `.` and `->` to access methods
- Using multiple format operators
- Finishing our `PositionReport` component.

Find the Transform Location

- Explore the API using `.` and `->`
- See if you can get the object’s location (X,Y,Z)
- Run and see if it prints on the Output Log
- Hint 1: It’s harder in XCode, complete isn’t fuzzy
- Hint 2: For the transform & location start with `Get`
- Hint 3: You will need to use `.ToString()`

v.4.10

Moving Objects With C++

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

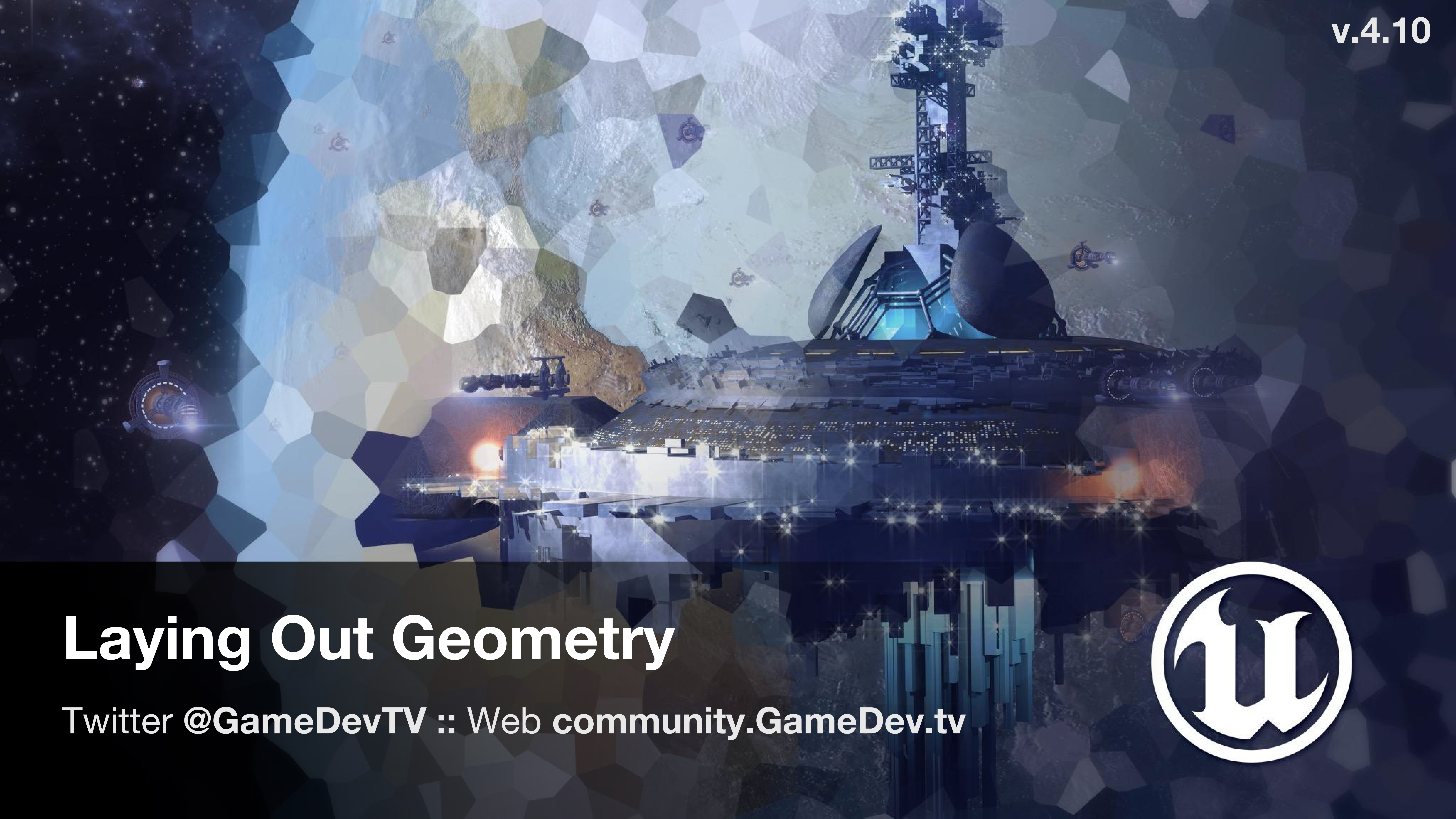
In This Video...

- A little more about the editor & temporary actors
- How to eject yourself from the possessed pawn
- Snapping objects to the floor (END key)
- Using the `FRotator` struct to represent rotation
- Use `SetActorRotation()` to rotate objects.

Access the Rotation

- Find the owning object as before
- Store it in a variable called `Owner`
- Get the type right (or use `auto`)
- Try and access the `Owner`’s rotation
- Hint: there are at least 2 ways.

v.4.10



Laying Out Geometry

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

In This Video...

- A brief intro of BSP “vs” Static Meshes
- Use **W, E, R** keys to translate, rotate, scale
- Make good use of grid snapping and quad view
- Hold **ALT + drag** translate to duplicate an object
- Hold **L** and double-click for temporary work **L**ight
- This is fiddly, try letting go of L and trying again.

Build Your First Room(s)

- Lay-out your room(s)
- Re-build the lighting
- Share in the discussions.

v.4.10

Applying Materials

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

v.4.10.
4

In This Video...

- A material is comprised of texture(s) and shader(s)
- Textures are image files, shaders are GPU code
- Unreal ships with some impressive examples
- Unreal has powerful material editing tools
- Applying materials to our room interior.

Customise Your Materials

- Apply materials as you see fit
- Play with their properties to see the effect
- Share your creations in the discussions!

v.4.10

Macros Starting with UPROPERTY

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

v.4.10.
4

In This Video...

- A macro is a programmed cut-and-paste
- This happens before the code is compiled
- Can unlock powerful functionality
- We don't get code complete as standard
- Can also create really weird build errors
- Expose `ATriggerVolume*` to the Details window.

Write Your First `UPROPERTY`

- Open your `OpenDoor.h` file
- Declare `ATriggerVolume* PressurePlate`
- Use the `UPROPERTY` macro but…
- … this time use the `EditAnywhere` parameter

https://docs.unrealengine.com/latest/INT/Programming/UnrealArchitecture/Reference/Properties/Specifiers/EditAnywhere/index.html

v.4.10

Using Trigger Volumes

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

v.4.10.4

In This Video...

- A trigger volume is a very versatile tool
- A 3D volume that detects things entering / leaving
- We’re going to use one as a pressure plate
- How we’re going to specify what can open doors
- Use `IsOverlappingActor()` on `ATriggerVolume`
- Polling vs using events.

v.4.10

Unreal's `PlayerController`

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

v.4.10.
4

In This Video...

- We’ve used `GetOwner()` to search “bottom-up”
- Now let’s use `GetWorld()` to search “top-down”
- Game Mode specifies the Default Pawn Class
- The Default Pawn is your “body”, is transient
- The Player Controller is your “mind”, persist
- PlayerController class has `GetPawn()`

Finish Your First Pressure Plate(s)

- Adjust size & position of your trigger volume(s)
- Link to appropriate door(s)
- Ensure all doors are “movable”
- Briefly test the gameplay.

v.4.10

Using Collision Volumes

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

v.4.10.
4

In This Video…

- Collisions volumes are also known as colliders
- These tell the physics engine what hits what
- A trigger volume just triggers code
- A collider actually has physics simulated
- Exploring how to add collision volumes
- Prevent players from passing through the door!

SM_Door is Ignored in SourceTree

- See “Line Tracing AKA Ray-casting”

Ensure You Can’t Escape

- Make sure there are no gaps in your colliders
- Test you can’t escape over walls
- Customise your collision volumes.

Using `GetTimeSeconds()`

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

v.4.10.
4

In This Video...

- Using `GetWorld()->GetTimeSeconds()`
- Making our game highly “play tunable”
- Re-factoring our code for simplicity
- Using a spotlight to provide “affordance”
- Play-testing to ensure the game is annoying!

Implement Door Close Delay

- Write some simple timing code
- Get the doors closing after a specified delay
- Play-test to ensure you can’t escape.

v.4.10

Grabbing System Overview

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

v.4.10.
4

In This Video...

- A quick look at the end result
- You try and think how it may be done
- I'll outline how we'll be doing it.

Write Your Ideas

- Use the knowledge you have already
- Would you use a component or inheritance?
- Hint: either could work, just hear yourself reason
- How may you know what to grab?
- What game object would you be working with?
- Share your ideas for discussion.

Grabbing System Overview

- We want to be able to lift the chair next
- We'll add a `Grabber.cpp` component to the player
- The player is a temporary actor, appears on play
- The Game Mode sets which Default Pawn to use
- Create Default Pawn & Game Mode Blueprints
- Specify our modified Default Pawn.

About GameMode

Anything from what inventory items a player starts with or how many lives are available to time limits and the score needed to end the game belongs to GameMode.

https://docs.unrealengine.com/latest/INT/Gameplay/Framework/GameMode/index.html

v.4.10

Modifying the Default Pawn Actor

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

v.4.10.
4

In This Video…

- Why Blueprint is helpful in this case
- How to make a Blueprint from the Default Pawn
- Note this Blueprint class inherits, an “is a” relation
- A Blueprint is like a template
- You make an “instance” in the scene
- Explore “instantiating” from Blueprint & modifying.

Try Making A Rugby Ball Pawn!

- Modify the Default Pawn somehow...
- ...scaling on one axis for example
- Create an instance by dragging into the world
- See how modifying instance doesn't change BP
- Revert your change.

v.4.10

Inherit Game Mode Blueprint

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

v.4.10.
4

In This Video…

- “Hard coding” means assets written into code
- The DefaultPawn_BP is an asset
- We want to be able to track changes to its name
- It is convenient to use Blueprint for this purpose
- Extending our C++ Game Mode with Blueprint
- Selecting the new DefaultPawn_BP

Make a Game Mode Blueprint

- Find the C++ Game Mode in the Content Browser
- Create a Blueprint class derived (inheriting) from it
- Set this as the Default GameMode in…
- Settings > Project Settings > Maps & Modes
- Make sure the game still runs the same.

v.4.10

Getting Player Viewpoint

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

v.4.10.
4

In This Video…

- Know where the player is looking
- Out-parameters can be confusing
- A way of marking-up out parameters
- Continuously logging player viewpoint.

Log the Viewpoint Every Tick

- Log the viewpoint position and direction every tick
- Hint: You may need to use `ToString()`
- Get used to working with different data types
- Give it at least 20 mins if you’re struggling
- Carry on watching for my solution.

v.4.10

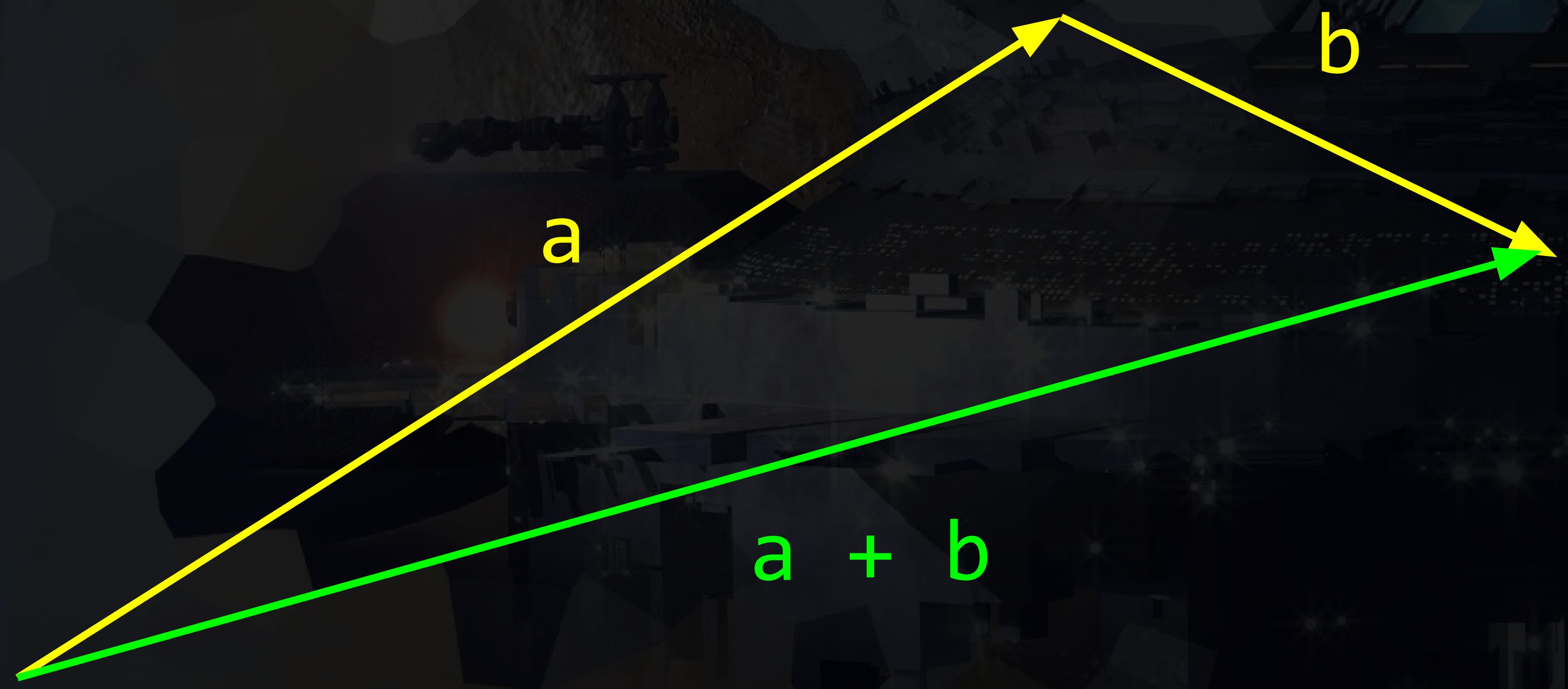
Using `DrawDebugLine`

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

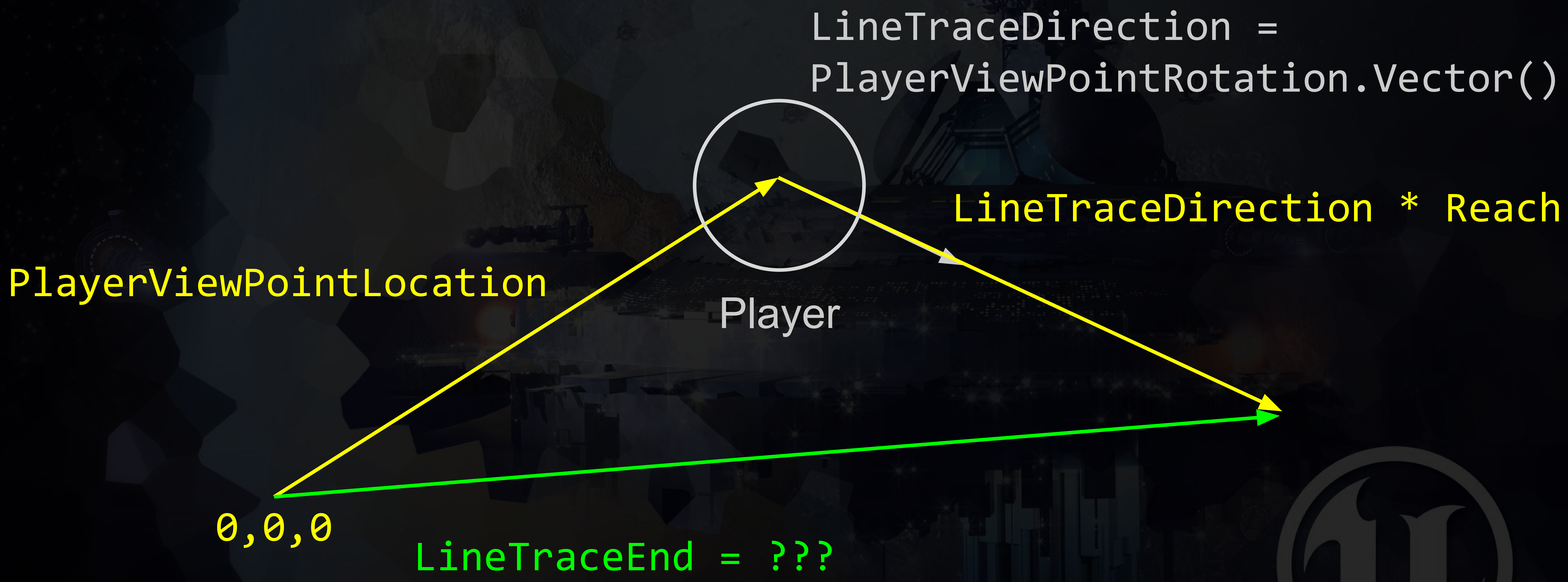
In This Video...

- How to add vectors
- Calculating our line trace end point
- Using debug functions for visualisation in Unreal
- Use `DrawDebugLine()` to visualise the vectors.

About Adding Vectors



Calculating LineTraceEnd



Calculate `LineTraceEnd`

- Create a private variable `float Reach = 100.f;`
- Calculate `LineTraceEnd`
- Test the debug trace, eject to visualise (F8)
- Share why it looks a square?

v.4.10

Line Tracing AKA Ray-Casting

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

v.4.10.
4

In This Video…

- Line tracing (AKA ray casting) is a very useful tool
- Imagine we shine a virtual laser into the world
- We can use different view modes to visualise
- Simulating physics sets the object channel.

Using Different View Modes

- Access via the view menu in 3D viewport
- By default this will be labeled as “Lit”
- Player Collision shows simplified meshes
- Visibility Collision shows complex meshes
- Hold Ctrl + Alt for more information.

Read About Collision Filtering

- Read Unreal’s blog post here…
- https://www.unrealengine.com/blog/collision-filtering
- The same rules apply to line tracing
- Share your understanding in the discussions.

v.4.10

LineTraceSingleByObjectType()

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

v.4.10.
4

In This Video...

- Meet references for the first time
- `LineTraceSingle` may be deprecated
- Build params inc. `FCollisionQueryParams`

https://docs.unrealengine.com/latest/INT/API/Runtime/Engine/Engine/UWorld/LineTraceSingleByObjectType

https://docs.unrealengine.com/latest/INT/API/Runtime/Engine/FCollisionQueryParams

Introducing References

- References are special pointers, denoted by `&`
- They cannot be `nullptr` or any other null value
- Once assigned they cannot be re-assigned
- Think of them like an alias
- You use them like the object they reference.

Log the Actor Hit

- Get an `Actor*` from `Hit`
- Perform a `->GetName()` on this actor
- Log the name to the console
- Test it works.

v.4.10

REFERENCES & POINTERS

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

In This Video…

- How references and pointers compare
- How to perform common operations in both
- What the & and * symbols means in context
- Challenge: Repoint and Rewrite
- When to use references over pointers?

Comparing Pointers to References

	Pointers	References
What is stored	Memory address	
Can be re-assigned (to another address)	Yes	No
Can be null	Yes (use `nullptr`)	No, must be initialised
Accessing contents	`*ActorPtr`	`ActorRef`
Accessing address	`ActorPtr`	`&ActorRef`
Changing the address	`ActorPtr = &Actor`	Not allowed
Changing the value	`*ActorPtr = Actor`	`ActorRef = Actor`

The & and * Symbols in Context

Context	When Using		When Declaring	
Code Examples	`CopyOfActor = *ActorPtr;` `ActorAddress = &Actor;`		`UActor* ActorPtr;` `UActor &ActorRef;`	
Symbol	`*`	`&`	`*`	`&`
Syntax	`*ActorPtr`	`&Actor` `&ActorRef`	`UActor*`	`UActor&`
Meaning	Contents at `ActorPtr`	Address of `Actor` or `ActorRef`	Pointer to `UActor`	Reference to `UActor`

To add insult to injury, these can have other meanings in other contexts, e.g. & for bitwise AND.

Repoint and Rewrite

- Repoint the pointer.
- How do addresses change?
- Rewrite the value at the pointer.
- Rewrite the value at the reference.
- What other values change?

Pointers or References?

- Golden Rule: Use references unless you can't.
- References are newer and safer.
- Pointers provide back-compatibility.
- Pointers are more powerful.
- They are also very dangerous.

v.4.10

Resetting Your Unreal Project

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

In This Video…

- What to do if your Unreal solution keeps crashing
- How to delete all temporary files
- The order in which to reset things.

Steps to Reset Your Project

1. **“Check-out” or “Reset”** - to a working commit
2. **Delete derived folders & files** - leave Config, Content & Source folders, and .uproject file
3. **Re-open Unreal** - from the launcher or .uproject*
4. **Generate your IDE project files**

** This re-creates generated.h files in Intermediate*

Try It Yourself

- Try resetting your solution, it’s important you’re confident how this works.
- If in doubt close everything first, then take a .zip of the whole folder as backup.
- Follow the steps on the previous slide
- Ask a question of other students if in trouble.

v.4.10

Using FindComponentByClass()

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

In This Video...

- What `FindComponentByClass()` does
- How to use it to find attached components
- Introducing angle brackets `<>` for generics
- Use `nullptr` to initialise your pointers
- Log a useful error if the component isn't attached.

Write Error Message

- Log at Error verbosity if no component found
- Write an error that helps the reader fix the issue
- Find and include the name of the object
- … in this case it’s the Default Pawn
- Temporarily remove component to test.

v.4.10

Introducing Input Binding

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

In This Video…

- Settings > Project Settings > Engine > Input
- Action mappings are used for on / off actions
- Axis mappings are used for analog values
- You can give players a way or re-mapping
- Many keys can bind to one action
- How to call a function on a key press or release.

Find the Input Component

- Create an appropriate private member
- Check for the component as Physics Handle
- Log a similarly helpful error if it’s not attached
- Don’t bother trying to remove to test this time.

v.4.10

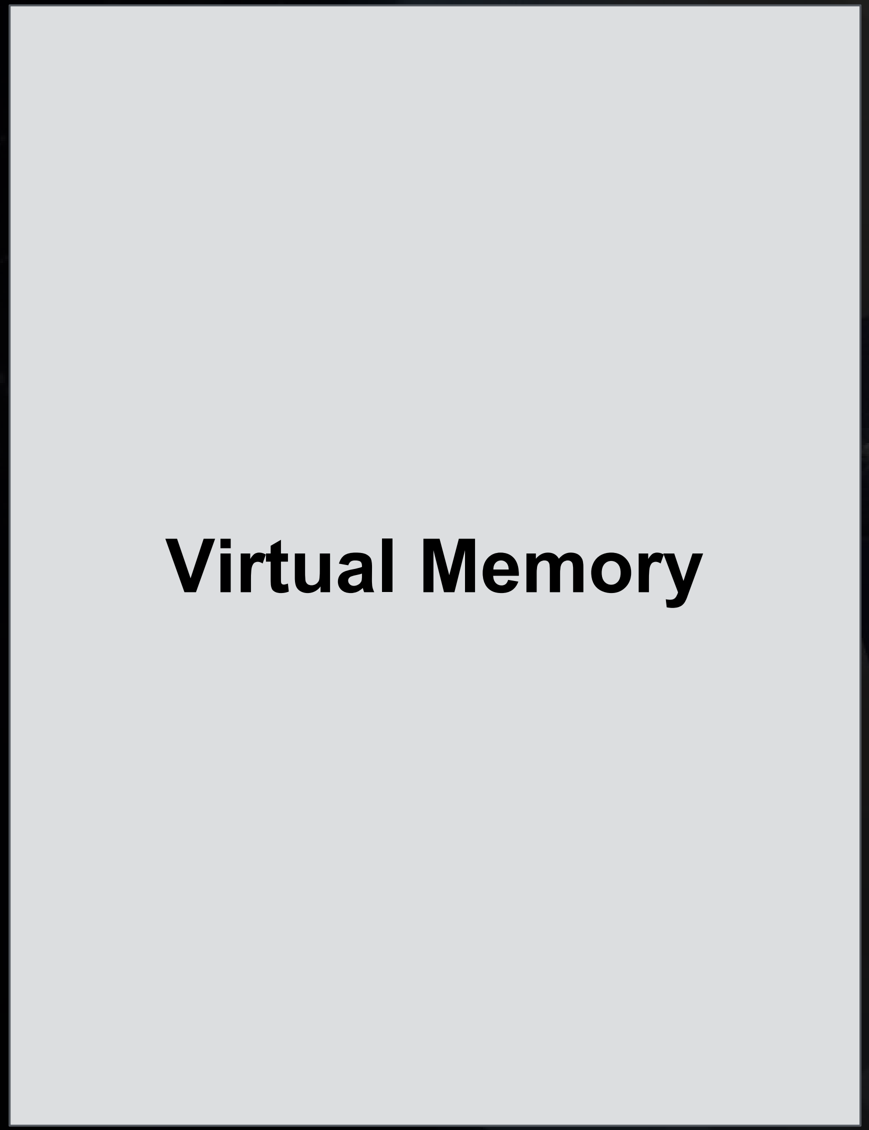
Accessors & Memory Layout

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

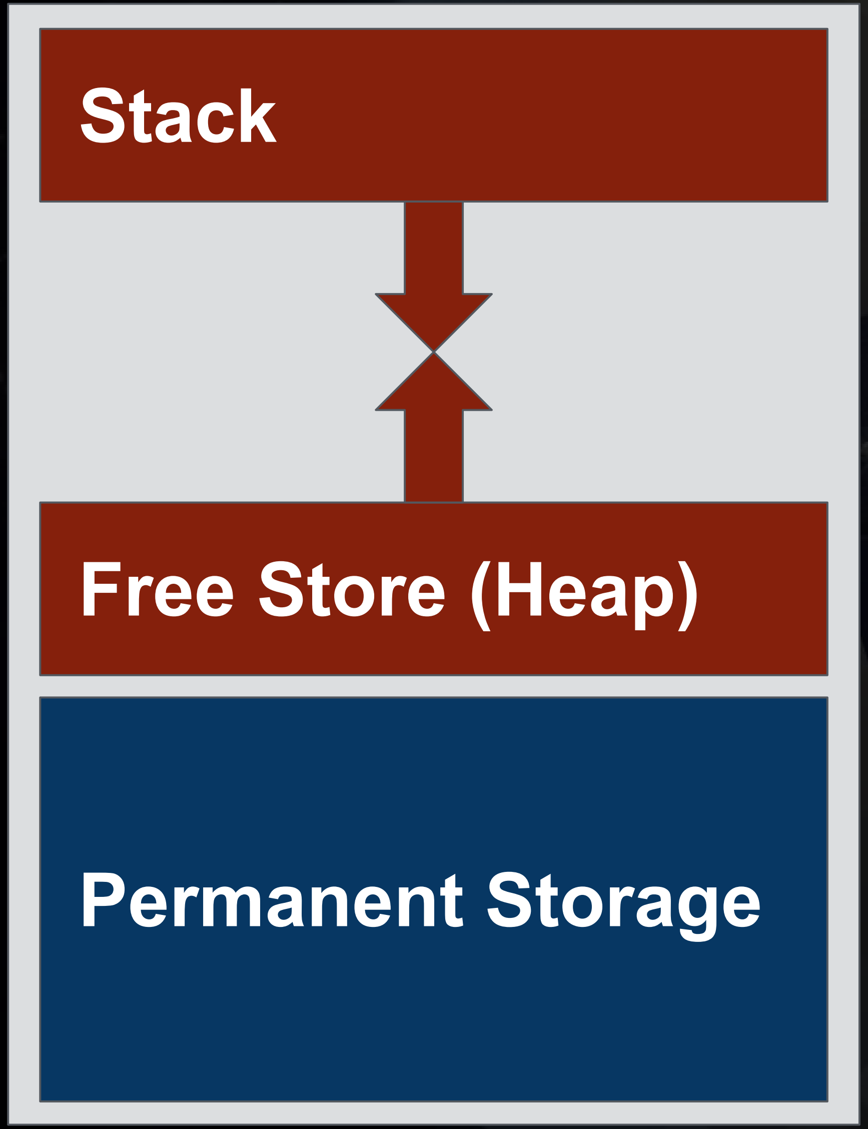
In This Video…

- How the arrow, dot and `::` accessors work
- Introducing virtual memory
- Introducing permanent storage, stack & heap
- Heap is also known as free store
- How accessor operators relate to memory
- Bind another input action.

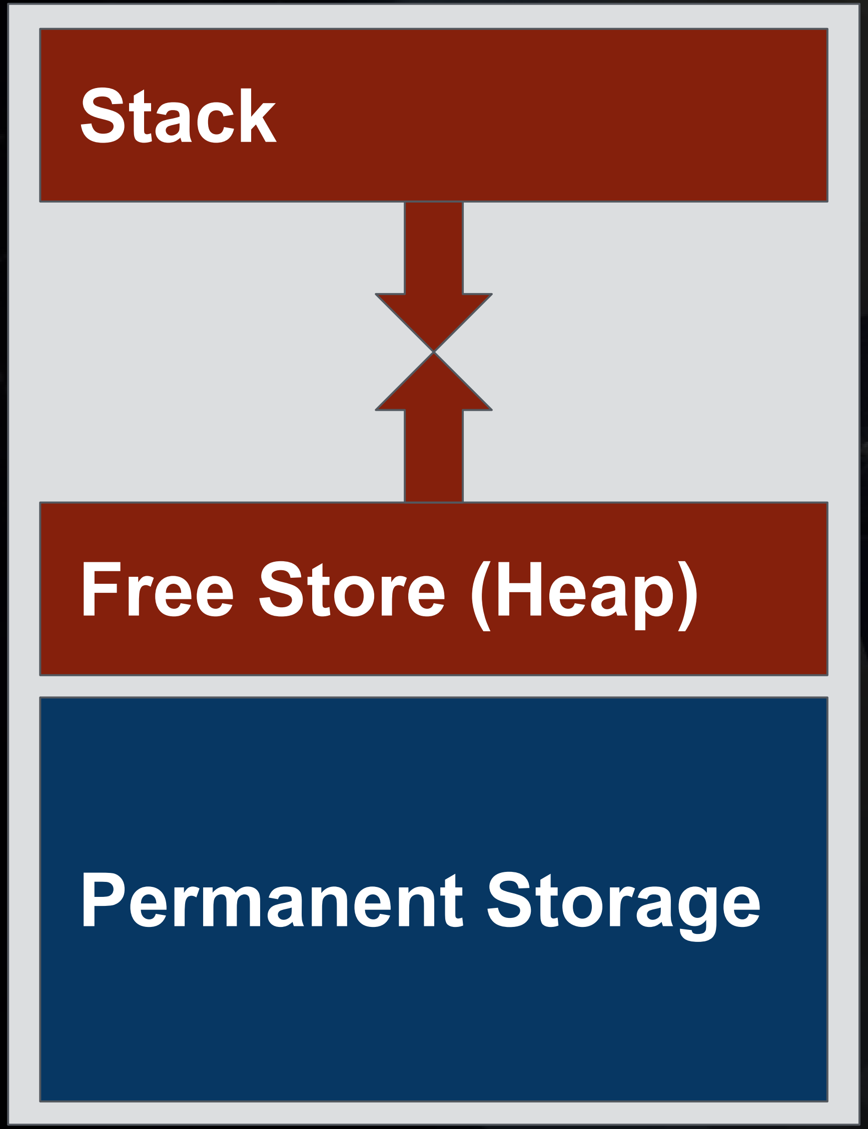
Accessors & Memory Layout



Accessors & Memory Layout



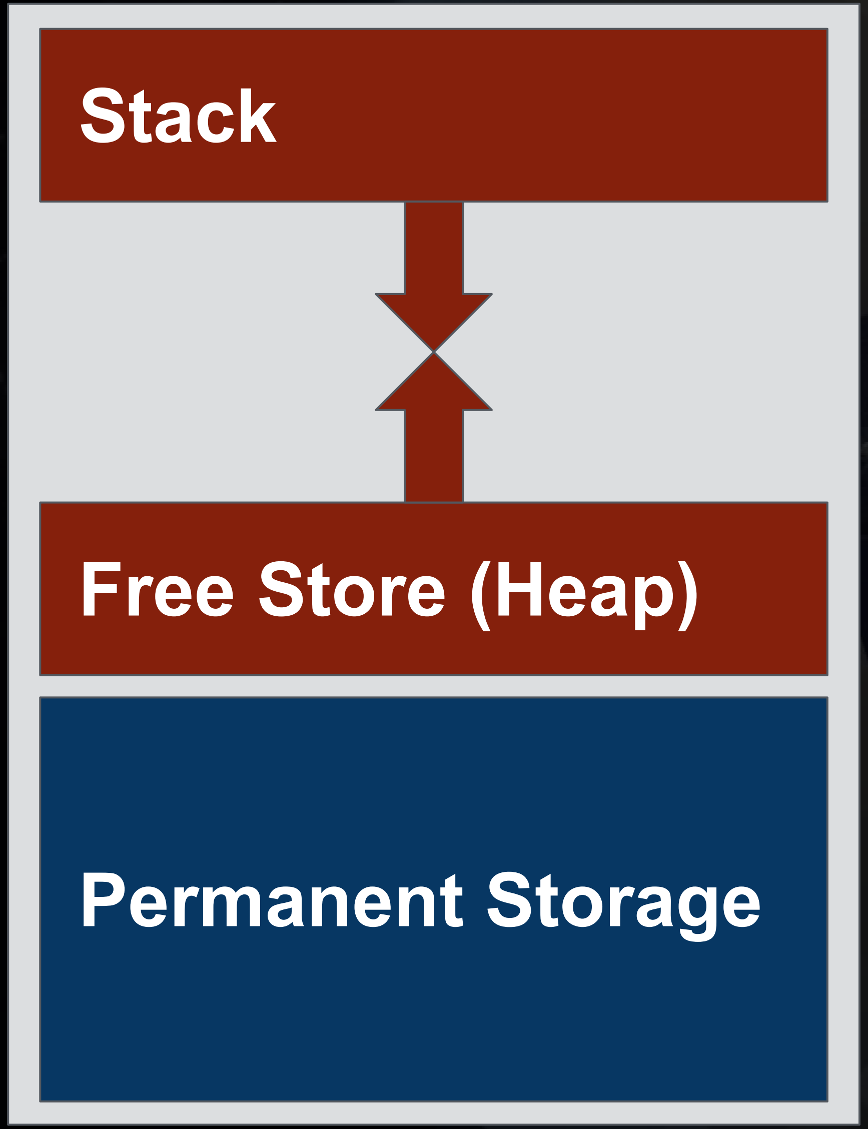
Accessors & Memory Layout



Left Term	Accessor	Examples

Class,
Enum,
Namespace
	`::`	`UGrabber::Grab`
		`EWordStatus::OK`
		`std::cout`

Accessors & Memory Layout



Left Term	Accessor	Examples

Instance or Reference	`.`	`MyGrab.Grab()`
		`MyBullCowCount.Bulls`
		`MyGrabRef.Grab()`

Pointer	`->`	`MyGrabPtr->Grab()`
		`MyGrabPtr->Reach`

Class, Enum, Namespace	`::`	`UGrabber::Grab`
		`EWordStatus::OK`
		`std::cout`

Create a `Release()` Method

- Follow the example of the grab binding
- The enum for release is `IE_Released`
- Log that the key has been released
- Test then jump with joy.

v.4.10

Reducing Code in “Hot Loops”

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

In This Video...

- A “hot loop” is code that get called often
- `TickComponent` is a good example, every frame
- Beware of code that you know will be called a lot
- Make it clear what happens every tick
- Refactor our code for speed...
- ...and make it ready for the physics handle.

Refactor your Code Too

- You can follow me through
- Or watch me first, then refactor at the end
- Or some hybrid, just get it so it’s clear to you
- Remember to run often, and commit when done.

Using Physics Handles

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

In This Video…

- Unreal provides a Physics Handle that’s ideal here
- The Physics Handle component docs are scant*
- Find an example of its use in the engine
- Get the physics handle working.

http://docs.unrealengine.com/latest/INT/API/Runtime/Engine/PhysicsEngine/UPhysicsHandleComponent

Red, Green, Refactor

- **Red** - It's not working (test failing)
- **Green** - It's working (ugly is OK)
- **Refactor** - Make it pretty (must still work!)

Then you repeat the sequence.

Find Example in the Engine

- Search engine code for `PhysicsHandle`
- Look for examples of it being used
- If that returns too much, try `GrabComponent(`
- See if you can find an example of its usage

https://docs.unrealengine.com/latest/INT/API/Runtime/Engine/PhysicsEngine/UPhysicsHandleComponent/GrabComponent

v.4.10

Refactoring Rules

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

In This Video...

- Using multiple getters for multiple return values
- Less lines of clear code is better (143 at start)
- Naming is really important, take the time
- Comment the “why”, don’t assume it’s obvious
- The “what” should be obvious...
- … but it can be helpful to add clarification.

Red, Green, Refactor

- **Red** - It's not working (test failing)
- **Green** - It's working (ugly is OK)
- **Refactor** - Make it pretty (must still work!)

Then you repeat the sequence.

Refactor Your Code

- Refactor your code again
- Yes it’s soon, but “clarity is worth fighting for”
- Commit once it’s done and it runs well
- Make it so clear you’ll remember in a year.

v.4.10

Introducing Unreal’s `TArray`

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

In This Video…

- A `TArray` is Unreal’s go-to container class
- Use to contain many elements of same type
- We’ll use to contain all actors on pressure plate
- Give our Default Pawn an eye-height and mass
- Making our pressure-plate based on total mass.

Read About `TArray`

- Skim-read the `TArray` documentation*
- Look out for the range-based `for` loop
- … particularly the pattern `for (x : y)`

*https://docs.unrealengine.com/latest/INT/Programming/UnrealArchitecture/TArrays

v.4.10

Iterating over `TArray` with `for`

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

In This Video...

- Using `auto&` as an auto reference type
- Automatically iterating over a `TArray`
- Pattern: `for (const auto* Iterator : Array)`
- How to find an actor’s mass
- Tweaking and testing our mass values.

Print the Name of Overlapping Actors

- Iterate over `OverlappingActors`
- For each actor found log their name
- Bonus: add their masses together and test
- Hint: class to find is `UPrimitiveComponent`

v.4.10

Debugging Game Issues

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

In This Video...

- Are you using source control? If not start now
- You can “binary search” commits quite fast
- For example 1024 commits takes max 10 tries!
- Think “what changed” and “possible side-effects”
- Remember you can eject with F8 during play.

Find and Eliminate “Drifting” Bug

- When did it come in?
- What feature did we recently enable?
- How does the pawn look when ejected?
- Hint 1: Enabling physics caused the issue
- Hint 2: Expand the “Constraints” section.

v.4.10

Managing Texture Tiling

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

In This Video...

- You may want to re-size objects (e.g. panels)
- Doing so will stretch the texture
- You can re-scale a few ways
- One way is in the material blueprint
- UV mapping because we ran out of letters!
- Using the TexCoord node in the material editor.

Adjust Your Textures

- Experiment with new textures
- Adjust the tiling as shown
- Share the results in the course.

v.4.10

Pointer Protection Process

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

In This Video...

- Horrible crashes when we follow a `nullptr`
- We must always check pointers before use
- When declaring always initialise to `nullptr`
- Look for `*` in your `.h` files to help find pointers
- Also check before every use and handle `nullptr`
- Sometimes we may choose not to, e.g. Owner.

Protect All Your Pointers

- Check the pressure plate pointer before use
- Log a helpful error if it’s null
- Test that it works
- Initialise any other uninitialised pointers
- Make sure all pointer usages are protected.

v.4.10

Exposing Events to Blueprint

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

In This Video...

- Sometimes Blueprint's the better choice
- For example defining our door swing as a curve
- We can create an event called `OnOpenRequest`
- Using `UPROPERTY (BlueprintAssignable)*`

https://docs.unrealengine.com/latest/INT/Programming/UnrealArchitecture/Reference/Properties/Specifiers/BlueprintAssignable

Set a Rotation & Test it Works

- Set a door rotation in Blueprint
- Test the game still plays the same
- Celebrate the fact you’re using C++ events in BP!

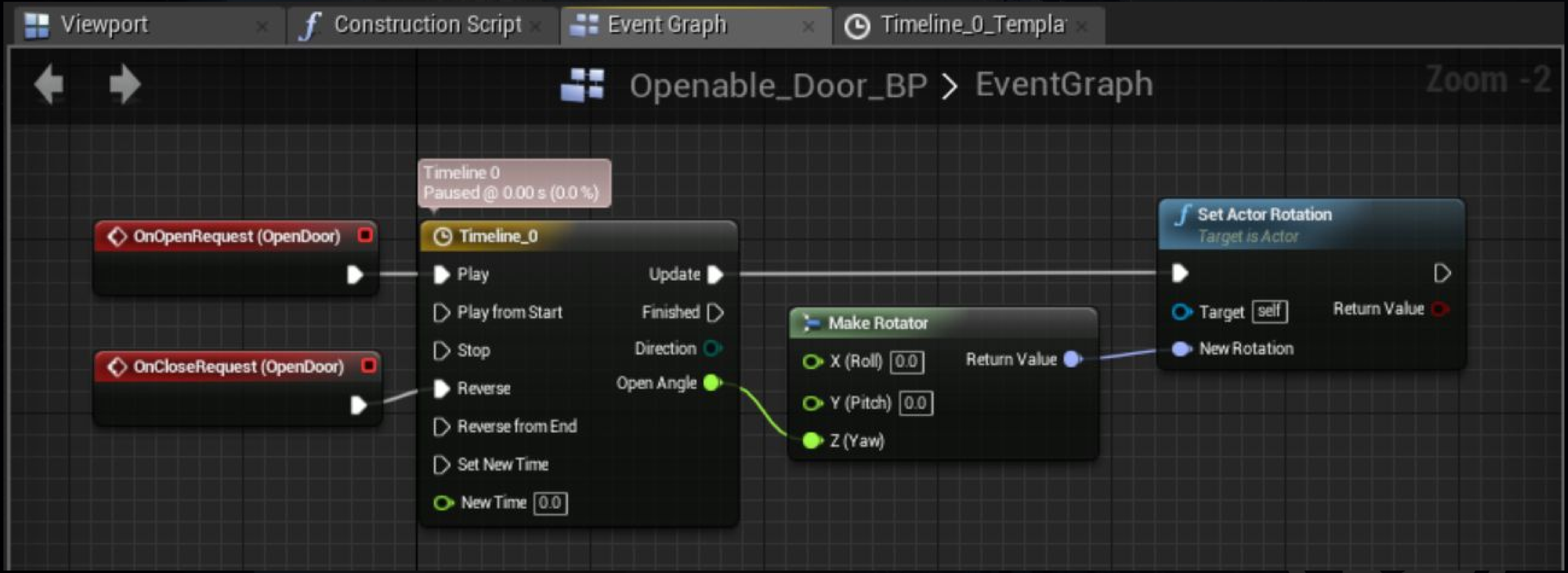
Using Blueprint Timeline

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

In This Video…

- The Timeline node in Blueprint has a curve editor
- This is ideal for defining our door movement
- How to use Timeline curves in Blueprint
- Setting rotation from a Timeline.

Final Blueprint Layout



View and comment on latest version online at http://bit.ly/UnrealSlides

Setup Your Door Movement

- See the final blueprint layout (I'll leave on screen)
- Set the curves to your taste
- Test you still can't leave the room
- You may need to adjust the room or curves.

v.4.10

Everything in its Place

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

In This Video…

- Using Blueprint has superseded some code
- It’s important there’s only 1 place per parameter
- Creating a 2nd event: `OnClose`

Create & Connect `OnClose`

- Rename the event class to `FDoorEvent`
- Rename `OnOpenRequest` to simply `OnOpen`
- Create a new `BlueprintAssignable` event
- Call this new event simply `OnClose`
- Wire `OnClose` into the “Reverse” pin in Blueprint
- Test the door now opens and closes.

v.4.10

Using Variables in Blueprint

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

In This Video…

- Not all doors have the same absolute rotation
- We want to store the door’s rotation at the start
- … then use this value to make a relative rotation
- We can use Blueprint variables for this
- Making doors that face any direction work.

Try and Combine Rotations

- See if you can finish the Blueprint off
- Look for a way of combining rotators
- Connect it all, test, debug and repeat
- Good luck!

SFX & Audio Clips

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

In This Video…

- We're going to trigger a simple sound in Blueprint
- Later in the course we'll use C++ too
- However we'll always reference our assets via BP
- How to trigger a 3D sound.

Get the Sound Working

- Get the sound FX playing on your own
- Consider making or finding other SFX
- Make a video and share in the course.

Section Wrap-Up

@UnrealCourse :: www.UnrealCourse.com

In This Video…

- Congratulations on another complete section
- You’ve learnt so much, look at the lecture titles
- Please carry-on a little on your own and share
- Attached are useful resources
- Start the next section as soon as you’re finished.

How to Delete a C++ Class

- YES - it should be easier than this!
- Delete the source files, and remove from project
- Rebuild the Visual Studio project files
- Delete Unreal's Binaries* folder
- Re-open the Editor and let it rebuild caches.

** Search for: "Unreal directory structure"*

Coming From Unity?

This Unreal document makes a helpful comparison between Unity and Unreal…

https://docs.unrealengine.com/latest/INT/GettingStarted/FromUnity/index.html?utm_source=launcher&utm_medium=ue&utm_campaign=uelearn