A Beginner's Introduction to Unreal Engine

Calem Bendell

McGill Game Dev

What is Unreal Engine (UE4/Unreal)?

A *massive* engine designed primarily for shooter games used to build videogames and other multimedia work

Regardless of your preferred engine, you should give Unreal a try!

You can learn from its more rigid structure, and playing with fire and things that explode builds character.

Our Schedule

- Look at other Unreal 4 Games (<=10 minutes)
- Compare Unreal and Unity (10 minutes)
- Talk about UE4 base classes (10 minutes)
- Do the coding and game making (the rest)

Games Built with Unreal

Unreal Engine 3

Mostly Designed for Shooter Games by Bigass Companies

Borderlands Series

Mass Effect

Batman Arkham Series

Revamped Engine Designed for Everyone to Make Anything

(or at least it’s trying to be)

Unreal Engine 4

Paragon

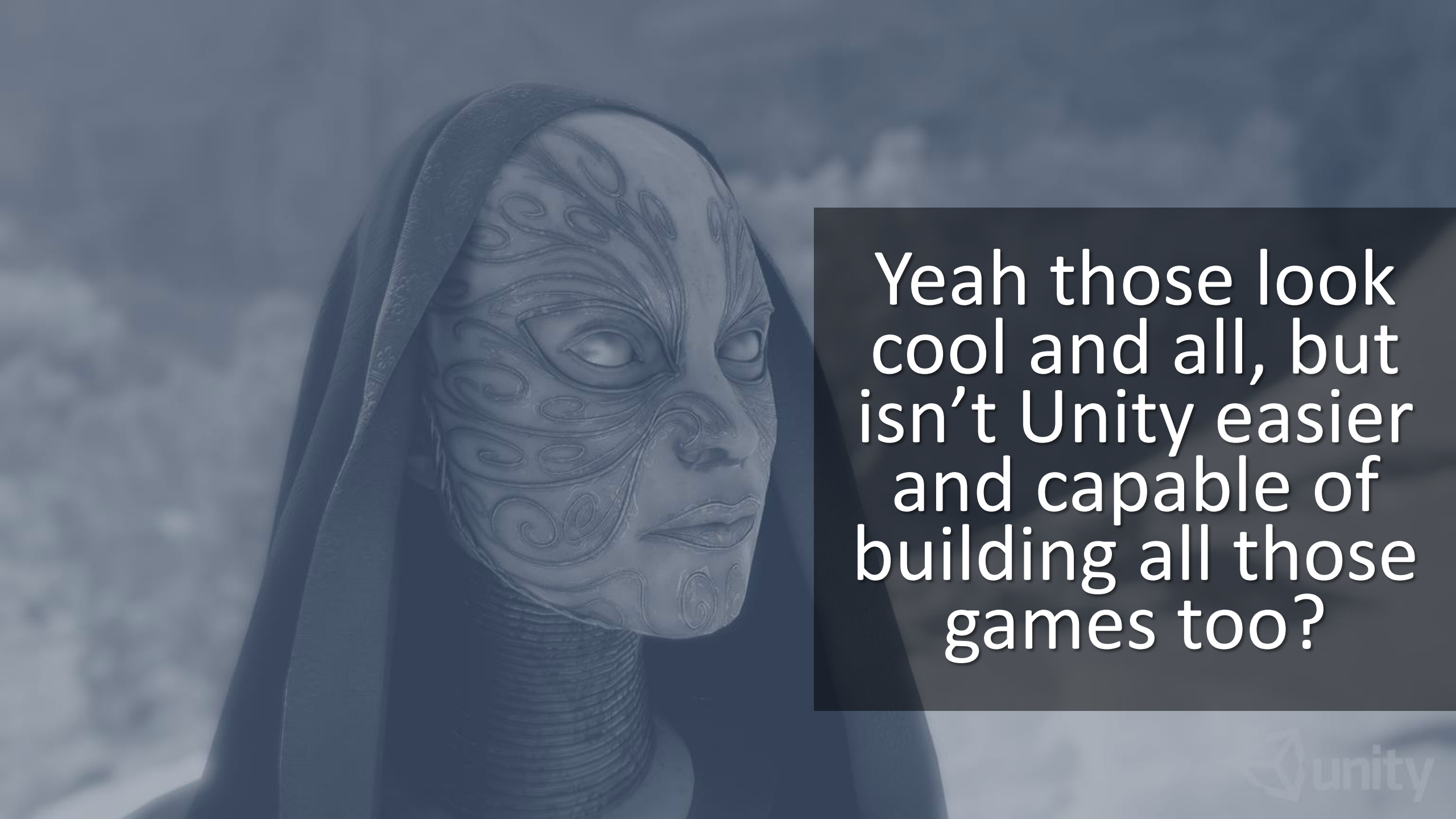
Fortnite

Dreadnought

Space Dust Racers

Ark

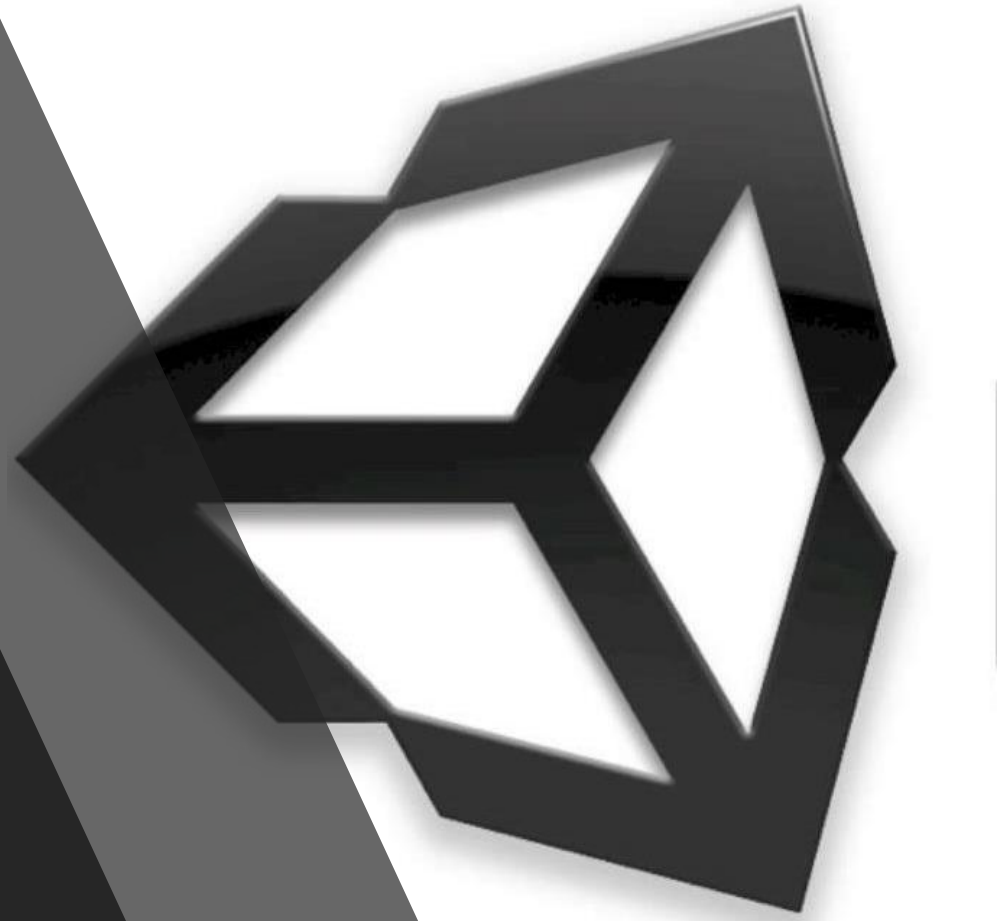
Shiness



Yeah those look cool and all, but isn’t Unity easier and capable of building all those games too?

Ummm… yes,
sort of.

A Very, Very Brief Comparison to Unity



Very brief, we have better shit to talk about!

Why to use UE4 (vs Unity)

- Full source code available! This is huuuuuuuuuuuge
- For experienced programmers, designed with heavy macro and reflection usage in mind
- Somewhat more advanced rendering
- Many more features out of the box than Unity
- Extending and editing the core engine is easy
- Arguably more rapidly developed than any other major engine
- More performant. But Unity is doing OK these days

Why to not use UE4 (vs Unity)

- Compile times are slow and C++ is definitely harder than C#/JS
- Performance occasionally tricky to achieve
- A much bigger initial learning investment
- The community isn’t as helpful
- Friendly documentation? Bah, go read the API and code!
- Packed with features that make the engine heavier but that you probably won’t use or know exist
- Requires a good development machine

The Summary *in general*

- Harder to learn
- You can manipulate the engine to do whatever you want, but the engine would rather make a shooter game
- But you'd be surprised how much is *easier* in Unreal 4 than other engines

Why I Use Unreal

- I found it easy to make a prototype in Unity but impossible to make something that felt polished or felt good to control
- Within a few days of starting to use Unreal, I had prototypes that controlled really well and looked better than they should have
- Most of the games I'm interested in making use humanoid characters or a decent bit of physics

The Learning Part

The Basic Classes

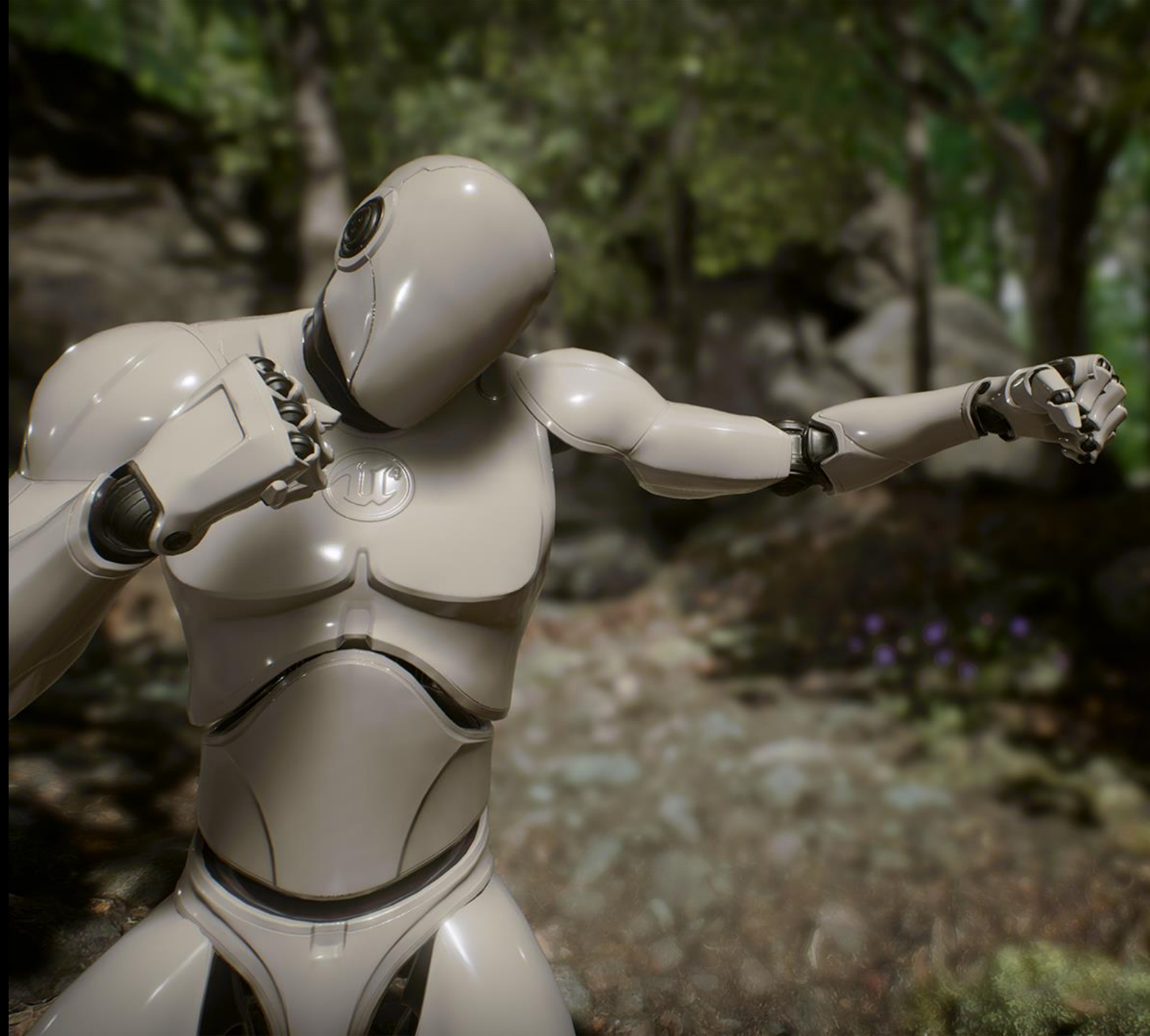
- Actors – objects that have a transform in the world space
- Pawn – the player
  - Character – a humanoid player
- Player Controller – player's way of controlling possessable pawns
- Game Mode – sets the rules
- Game State – syncs things between players

Actors

- The basis of just about everything in an Unreal game
- Inhabit the world
- Can have components to them, like meshes or collision boxes or ai perception

Pawns

- Actors that can be controlled by a player

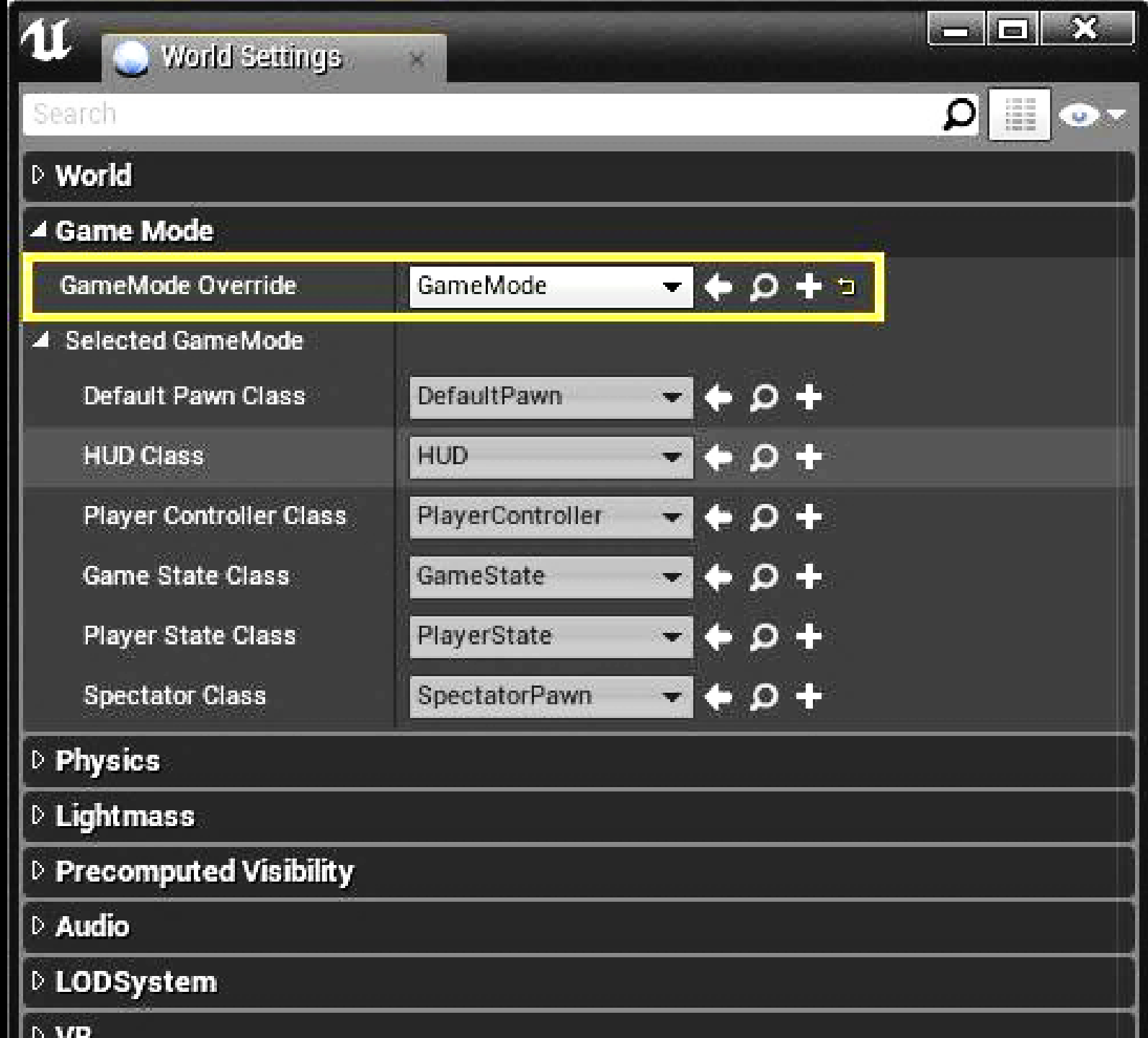


Player Controller

- Defines how the player controls the pawn or even simply tracks which pawn is being controlled

Game Mode

- How many players/spectators
- Where and when to spawn
- Pausing
- Entering new levels



Game State

- Keeping score
- Current states of players
- Are we paused right now?!

And many more

But let’s leave those for another day, for now

Some of the Most Difficult Parts of Most Gaming Engines

- Defining interactions between actors
- Defining default behaviours or components and overriding them
- Handling player input
- Spawning things, especially hierarchies

We’re going to be using mostly C++

Blueprints make a lot of things easier, so we’ll maybe do a little bit of stuff from Blueprints if we have the time, but C++ is where a lot of the power of UE4 is at!

Defining Actors

- Make a new class by going to the “Create new C++ class” menu
- Choose a class to extend, in our case “Actor”
- Give it a name
- Once the code stub is generated, we can add properties and functions!

UPROPERTY

- A UPROPERTY can be viewed in the editor
- UPROPERTY is a macro that creates extra data used for introspection and Blueprint calls
- Example:

```
UPROPERTY(EditAnywhere)
 float MaxSpeed = 100.f;
```

Components

- Actors can have components such as UPointLightComponent, UBoxComponent, UStaticMeshComponent, etc.
- For rolling a ball around, we know we'll want a static mesh to roll around
- For shits and giggles, we'll add a pointlight to make the area around the ball glow

```
UPROPERTY(EditAnywhere)
  UStaticMeshComponent* BallMesh;
UPROPERTY(EditAnywhere)
  UPointLightComponent* Light;
```

Making a constructor for our actor

- We've given the actor some component pointers
- Now we need to instantiate them
- {VariableName} = CreateDefaultSubobject<{Type}>(TEXT("Name"))

BallMesh = CreateDefaultSubobject<UStaticMeshComponent>(TEXT("Ball");

RootComponent = BallMesh; // this is the "base" of our actor

BallMesh->SetRelativeRotation(Fvector(0.f, 0.f, 90.f)); // why not

UFUNCTION

- Like UPROPERTY, defines a function that can be called from Blueprints or a function that can be defined in Blueprints but used from C++

```
UFUNCTION(BlueprintCallable, Category = Control)
  void RollRight(float AxisValue); //!< rolls ball left and right with axis input
```

Moving To Using Visual Studio

The rest of this workshop will be done directly in UE4 (4.12 – 4.13 is fine) and Visual Studio 2015 (express is fine)