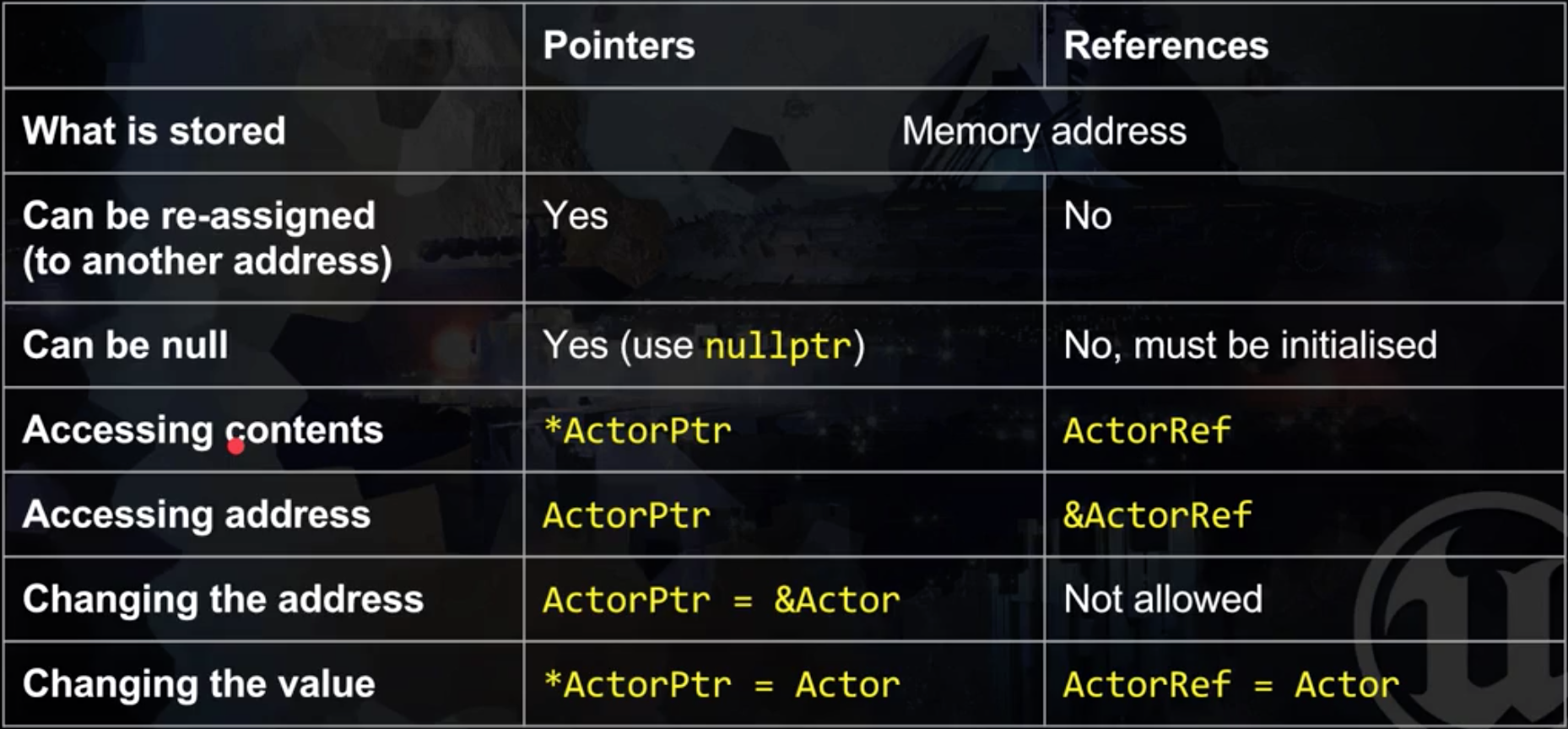
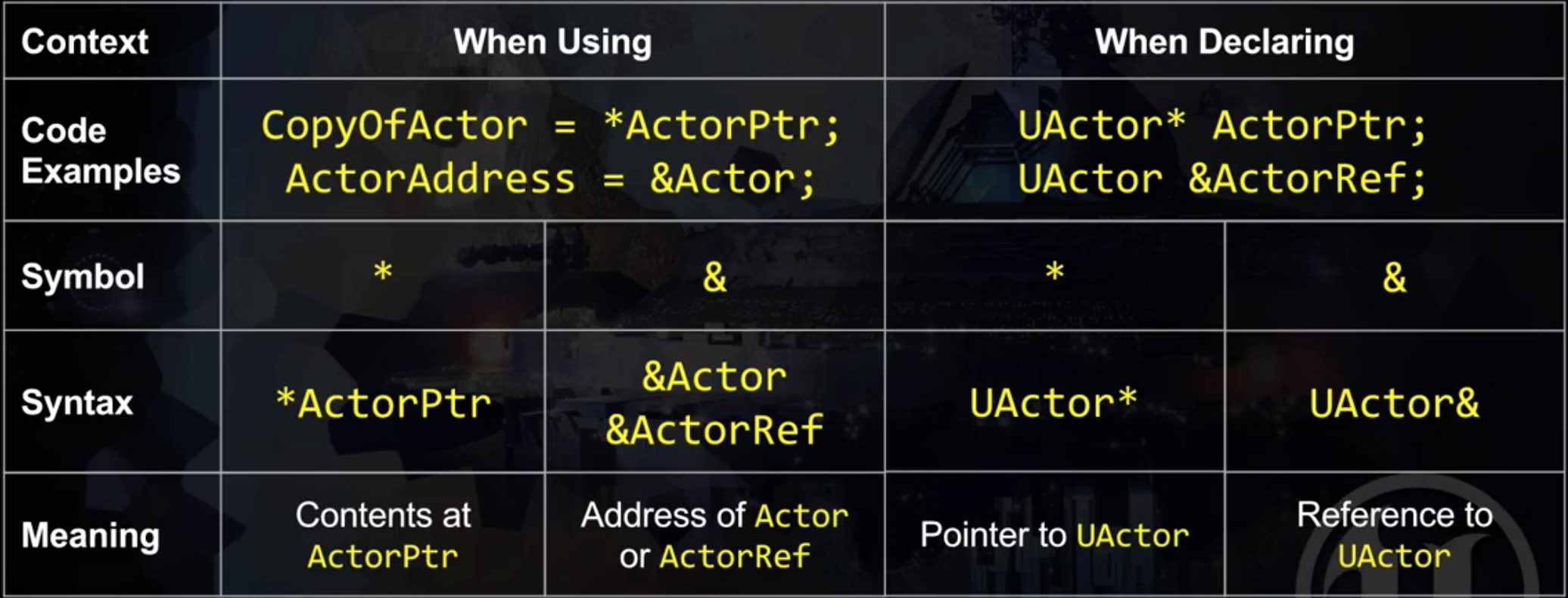
*Unreal Engine C++ Notes*

* Float values must be postfixed by an f: 2.08f
* In a header file named xyz.h, the #include “xyz.generated.h” must be the last include.
* Cast<ASomeActor>(GetPawn()); to cast GetPawn().
* To add blueprint functionality to a C++ class, make a blueprint class which is a child class of the C++ class.
* Including virtual in a method declaration allows it to be overridden by its children.
* To extend a parent method:  
  virtual void BeginPlay override; // (header file)  
  void ASomeActor::BeginPlay() // (c++ file)  
  {  
   Super::BeginPlay();  
   …  
  }
* Overriding Tick:  
  virtual void Tick(float DeltaTime) override; // (header file)  
  void ASomeActor::Tick(float DeltaTime) // (c++ file)  
  {  
   Super::Tick(DeltaTime);  
   …  
  }
* A component has method TickComponent, rather than Tick. Signature is different, too.
* To log: UE\_LOG(LogTemp, Display/Warning/Error, TEXT(“Message”));
* For dynamic messages: TEXT(“String is %s and %s”), \*(GetString1()), \*(GetString2())
* C++ Polymorphism
  + Overloading (ad-hoc): different implementation per signature
  + Subtype (runtime polymorphism): parent pointers pointing at children instances
  + Parametric: one implementation, multiple signatures
* Postfix const if method doesn’t modify anything.
* Prefix auto to a variable declaration to automatically deduce its type.
* Can use GetWorld() to …
* a.b (period) vs a->b (arrow)
  + AActor\* SomeActor;
  + \*SomeActor de-references the pointer.
  + (\*SomeActor).GetName(); follows memory address and then accesses method.
  + SomeActor->GetName(); follows and accesses in a single operator.
  + The two are equivalent, but arrow is shorter.
* After getting a pointer, protect it (quit early if null):  
  if (!pointerThing)  
  {  
   return nullptr;  
  }
* Outparameters (Used in engine, but clunky to use otherwise?)
* References (&) vs Pointers (\*)  
    
  
* Example transforms syntax: GetOwner()->GetTransform().GetLocation().ToString()
* Common types: FString, FVector, FRotator, AActor, …
* UPROPERTY: Use it to make fields accessible in the Editor  
  UPROPERTY(VisibleAnywhere/EditAnywhere)  
  float ExampleFloat = 90f;
* Generics
* Physics handle
  + To initialize physics handle: UPhysicsHandleComponent\* PhysicsHandle = nullptr;
  + PhysicsHandle->GrabComponent(ComponentToGrab, NAME\_None, ComponentToGrab->GetOwner()->GetActorLocation(), true);
  + PhysicsHandle->ReleaseComponent();
  + PhysicsHandle->SetTargetLocation(SomeVector)
* Using find component by class: GetOwner()->FindComponentByClass<UPhysicsHandleComponent>()
* To access enum value: UGrabber::Grab