Unreal Classes

Tim Sweeney
Epic MegaGames?, Inc.
http://www.epicgames.com/

Audience: Level Designers, UnrealScript? Programmers, C++ Programmers.
Last Updated: 07/21/99

About this document

This is a quick attempt to describe the most important classes and variables in Unreal. It is currently very sparse but will be expanded over time.

Engine.Object

Purpose:

Serves as the base class of all objects in Unreal. All objects inherit from Object.

Variables:

- **Parent:** The object's parent object (for scoping).
- **ObjectFlags:** The object's flags, described in the Packages document.
- **Name:** The object's name.
- **Class:** The object's class.

Functions:

- **Log**: Writes a message to the log file, usually \Unreal\System\Unreal.log.
- **Warn:** Writes a script warning to the log file, including the current script and function.
- **Localize**: Returns a localized (internationally translated) string from a package's .int file.
- **GotoState**: Sets the object's current state, None means no state. If no label is specified, the Begin label is gone to.
- **IsInState**: Returns whether this object is in the specified state.
- **GetStateName**: Returns the name of this object's current stae, None if none.
- **Enable**: Enables a probe event. The only functions which work with Enable and Disable are: Spawned, Destroyed, GainedChild?, LostChild?, Trigger, UnTrigger?, Timer, HitWall?, Falling, Landed, ZoneChange?, Touch, UnTouch?, Bump, BeginState?, EndState?, BaseChange?, Attach, Detach, ActorEntered?, ActorLeaving?, KillCredit?, AnimEnd?, EndedRotation?, InterpolateEnd?, EncroachingOn?, EncroachedBy?, FootZoneChange?, HeadZoneChange?, PainTimer?, SpeechTimer?, MayFall?, Die, Tick, PlayerTick?, Expired, SeePlayer?, EnemyNotVisible?, HearNoise?, UpdateEyeHeight?, SeeMonster?, SeeFriend?, SpecialHandling?, BotDesireability?.
- **Disable**: Disables a probe event.
- **GetPropertyText**: Converts the value of an arbirary variable to text.
- **SetPropertyText**: Sets the value of an arbitrary variable from text.
- **SaveConfig**: Saves the current values of all "config" variables to the Unreal.ini file.
- **ResetConfig**: Resets the values of the "config" variables to the originals in the Default.ini file.

Static Functions:

- **ClassIsChildOf**: Returns whether one class is a subclass of another class.
- **GetEnum**: Returns the nth element of an enumeration.
- **DynamicLoadObject**: Tries to load an object from a file, and returns it.

Events:

- **BeginState**: Called when a new state is entered.
- **EndState**: Called when the current state is ended.

Engine.Actor

Functions:

- **Error**: Causes a critical error; exits the engine.
- **SetCollision**: Changes the actor's collision flags.
- **SetCollisionSize**: Changes the actor's collision size.

- **Move**: Moves the actor by the specified displacement vector, handling collision (based on the collision flags), Touch and Bump notifications.
- **MoveSmooth**: Like move, but smoothly brushes against walls.
- **SetLocation**: Teleports the actor to a new location.
- **SetRotation**: Sets the actor's new rotation.
- **SetBase**: Sets the actor's Base. A base of None means that the actor moves alone; setting the base to another actor in the world causes this actor to move and rotate along with its base. An example of using a base is standing on a moving platform.
- **SetOwner**: Sets this actor's owner.
- **IsA**: Return whether this actor belongs in a named class.
- **PlayAnim**: Plays a named animation sequence in the actor's Mesh once (use the Mesh Viewer in UnrealEd? to see the animation sequence names for a given mesh). The optional Rate scales the animation's default rate. If a nonzero TweenTime? is specified, the animation is first tweened from whatever is currently displayed to the start of the named animation sequence, before the animation sequence plays. When the animation playing completes, it stops and calls your optional AnimEnd?() event and causes any latent FinishAnim?() calls to return.
- **LoopAnim**: Like PlayAnim?, but loops the animation sequence over and over without end. As with PlayAnim?, the AnimEnd?() is called at the end of the sequence, and FinishAnim?() calls return at the end of each iteration of the loop.
- **TweenAnim**: Tweens from whatever animation is currently being displayed, to the start of the specified animation sequence, then stops and calls AnimEnd?() and releases any latent FinishAnim?() calls.
- **IsAnimating**: Returns whether the actor's mesh is currently animating.
- **GetAnimGroup**: Returns the group name of the specified animation sequence.
- **SetPhysics**: Sets the actor's current physics mode.
- **BroadcastMessage**: Sends a text message to all players.

Latent Functions:

- **Sleep**: Waits for a certain (fractional) number of seconds to pass, then continues.
- **FinishAnim**: Waits for the currently playing or looping animation to reach the end of the sequence.
- **FinishInterpolation**: Only relevent with the physics mode PHYS_Interpolating. Waits until the next interpolation point is reached.

Iterator Functions

- AllActors: Iterates through all actors in the level with the specified tag; if you don't specify a tag, it iterates through all actors in the level. This is pretty slow if you specify a tag, and very slow if you don't.
- ChildActors: Iterates through all actors owned by this actor.
- BasedActors: Iterates through all actors directly based on this actor.
- TouchingActors: Iterates through all actors touching this actor. This is fast.
- TraceActors: Iterates through all actors hit by a trace along a line with an optional collision extent. This is pretty fast.
- RadiusActors: Iterates through all actors within a specified radius. This is fast.
- VisibleActors: Iterates through all actors visible to this actor (using a simple line trace, rather than exact mesh-to-mesh visible). This is very, very slow.
- VisibleCollidingActors: Iterates through all actors visible to this actor which have collision enabled; this is much faster than VisibleActors?.

Events:

- **AnimEnd**: Called when the currently playing or looping animation reaches the end of the sequence.
- **Spawned**: Called when an actor is spawned during gameplay. Never called if the actor is preexisting when a level is loaded.
- **Destroyed**: Called immediately when an actor is destroyed. This event is not delayed like Java's Finalize() event is.
- **Expired**: Called immediately before Destroyed() when an actor's LifeTime? has counted downward past zero.
- **GainedChild**: Another actor has done a SetOwner?() to this actor.
- **LostChild**: An actor owned by this actor has been destroyed or done a SetOwner?() to another actor.
- **Tick**: Called each iteration of the game loop; varies depending on frame rate. DeltaTime? indicates how much time has passed.
- **Trigger**: This actor has been triggered by another actor Other, optionally caused by a pawn EventInstigator?.
- **UnTrigger**: This actor has been untriggered by another actor Other, optionally caused by a pawn EventInstigator?.
- **BeginEvent**: A simple protocol to indicate that this actor called a Trigger() function that began a complex chain of events. This enables switches to be coordinated with doors intelligently, i.e.: You press a switch. The switch moves in. It triggers a door which calls your BeginEvent?() then slowly opens. The door then calls your EndEvent?(). Now you move the button back out to give the user feedback that the button is finished with what it was doing.
- **EndEvent**: The event specified by BeginEvent? has ended.
- **Timer**: If you call SetTimer?, this Timer() function is called each time the timer has counted down from your time value.
- **HitWall**: This actor ran into a wall while moving.
- **Falling**: This actor has transitioned into PHYS_Falling.
- **Landed**: This actor has transitioned out of PHYS_Falling.
- **ZoneChange**: This actor has entered a new zone.
- **Touch**: This actor has begun touching (interpenetrating) another actor.
- **UnTouch**: This actor has stopped touching (interpenetrating) another actor.
- **Bump**: This actor has bumped into an impenetrable actor.
- **BaseChange**: This actor's base has changed.
- **Attach**: Some other actor has set its base to this actor.
- **Detach**: Some other actor based on this one is no longer based on this actor.
- **KillCredit**: This actor has successfully killed another actor.
- **SpecialHandling**: ?

- **EncroachingOn**: This actor is now encroaching on (overlapping with) another actor as a result of moving and it may return true to abort the move, or false to continue the move.
- **EncroachedBy**: This actor has been successfully encroached by (overlapped by) another actor. This is where players may opt to telefrag themselves or ignore the encroach.
- **InterpolateEnd**: Called when Physics==PHYS_Interpolating and the next interpolation point has been reached.
- **KilledBy**: Called when killed by another actor (killed just means dead, it doesn't mean destroyed).
- **TakeDamage**: A certain amount of damage has been imparted on this actor.
- **PreTeleport**: Called before this actor is teleported by a teleporter.
- **PostTeleport**: Called after this actor is teleported by a teleporter.

Physics Modes:

- To be written.

Game Startup:

When programming new actor scripts, you need to be wary of the order in which actors that pre-exist in a level (at load time) are initialized, and the order in which new actors are initialized when they are spawned. The exact order of initialization is:

1. The actor object is created and its variables are initialized to their default values (if spawning an actor), or loaded from a file (if loading a pre-existing actor).
2. The actor's location, rotation, and owner are set.
3. If the actor is being spawned, its Spawn() event is called.
4. The actor's PreBeginPlay? () event is called.
5. The actor's BeginPlay? () event is called.
6. The actor's zone is set, and any ZoneChange? messages are sent to the actor, and any ActorEntered? messages are sent to its zone. Prior to this point, the actor is not in a valid zone, and you must not call the Move, MoveSmooth?, SetLocation?, or SetRotation? functions, or any functions which call them.
7. The actor's PostBeginPlay? () event is called.
8. The actor's SetInitialState? () event is called.
9. If the actor hasn't set its base yet, its base is set.