Technical Design Document

Engine
Unreal Engine 4.18.1

Coding standards and style guide
Epic Coding Standard
C++ Coding Standards
Gamemakin UE4 Style Guide

Architecture
Before implementing a feature, check if it comes with the engine. If it does, prefer to use that version unless you have a good reason not to. Using Unreal's features saves time, and they are probably better anyway.
Gameplay programming should be done in C++. Once the final C++ class in an Actor-based hierarchy has been written, it can be derived in Blueprint to add meshes, materials, animation Blueprints, etc., and be placed in the level or spawned at runtime. In other words, Blueprint classes are the final level of decomposition - they should always be based on custom C++ classes, they should never be base classes for other Blueprint classes, and should only add gameplay logic or variables for prototyping purposes.
Every C++ class receives its own folder, with the same name as the class, located in the folder of its base class.
The Content directory should mirror the Source directory where possible.

Construction
Header files should have one section for public members, one section for protected members, and one section for private members, in that order. Within each section, members should be declared in the following order: typedefs, enums, constants, constructors, the destructor, methods, data members with UPROPERTY macros, and data members without UPROPERTY macros.
Use the bindings in DefaultInput.ini instead of hard-coding inputs. This will allow developers to edit an input in one place, and have that edit be reflected throughout the entire game. This will also enable players to customize their controls.
Use these TArray optimizations.

UPROPERTY specifiers	
Specifiers	**Usage**
EditInstanceOnly	EditInstanceOnly exposes properties for editing in the details panel for instances (i.e., actors in the world outliner). This specifier is used if and only if the property will have different values on different instances.
EditDefaultsOnly	EditDefaultsOnly exposes properties for editing in the details panel for archetypes (i.e., blueprints in the blueprint editor). These specifiers are only used for UASSET references.
No UPROPERTY specifier	If a non-UASSET property (int, float, string, or enum) will have the same value across all instances, it should not receive a UPROPERTY specifier. Design constants are deserialized, and other constants (such as AI data, UX data, and FNames for collision profiles, sockets, and blackboard keys) can be found in the Constants namespace.

Serialization

At the end of every game session, a timestamped report folder will be added to C:/ContrabandLogs. The folder will contain:

- A text report containing informational logs and warnings (e.g., missing UASSET references). If a report contains any warnings, a "_w" will be appended to the file and folder names. At the end of every player modeling interval, the player's current status (including level, health, location, checkpoint, objective, and inventory) is appended to the log as well.
- A CSV report of gameplay analytics over time for data visualization.

Player Modeling

Statistics tracked
ProjectilesFired
ProjectileHits
DamageDealt
DamageTaken
Deaths
Performance
Accuracy
Kills
FirearmsThrown
BoomerangThrows
CheckpointsTriggered
ObjectivesComplete
LevelsComplete

Statistic contexts
Interval
Moving average
Current life
Checkpoint
Objective
Level
Campaign

Dynamic Multicast Delegates

Delegate	Actor
OnNewObjective	ContrabandGameModeBase
OnObjectiveComplete	ContrabandGameModeBase
OnBanner	ContrabandHUD
OnTutorial	ContrabandHUD
OnCheckpoint	ContrabandHUD
OnAgentDamaged	Agent
OnAgentKilled	Agent
OnDamagedOtherAgent	Agent
OnKilledOtherAgent	Agent

OnStartedTargetingObject	Player
OnStoppedTargetingObject	Player
OnFiredFirearm	Agent

Cheats

Cheat	Console command	Parameters
Toggle god mode	God	
Toggle infinite ammo	Ammo	
Toggle cursor	Cursor	
Damage target, or player if player has no target	Dmg	Amount of damage to inflict
Heal target, or player if player has no target	Heal	Amount of health to heal
Kill target, or player if player has no target	Kill	
Enable debug until code is recompiled or editor is closed	Debug	
Set the player modeling debug statistic	DebugStat	Name of statistic to view
Spawn the nearest spawner's default actor to spawn	Spawn	
Reload audio engine soundbank	Reload	
Restart level	Restart	
Change the difficulty setting	Difficulty	New difficulty index
Enable autoplay	Autoplay	
Complete the current objective	CompleteObjective	

Level Guidelines

Technical guidelines
Objects that are the same in every level should be created at runtime. This will promote consistency between levels, and reduce clutter in the world outliner.
Use folders in the world outliner.
Levels should be built up from 0 on the Z-axis. Feedback systems will check the player's Z-position, and will behave strangely in levels built high or low.
Confirmation must be given before the player permanently leaves an area. This can be explicit (a prompt, as in Wolfenstein) or implicit (walking off an edge that is obviously a point of no return).
Levels should be designed to support unobtrusive level streaming.
Level scripting should be implemented using Level Blueprints.
Use level scripting sparingly. If you find yourself reusing level scripting (i.e., copy/pasting Level Blueprints), you can have technical team make a class for you.

Design guidelines
Levels should emphasize combinations of indoor and outdoor environments (as in Halo).
Levels should emphasize contrast between dark and lit areas.
Levels should emphasize environmental storytelling.
Levels should feel like believable, plausible spaces (as in Dark Souls and Fallout: New Vegas).

Levels should show or imply an abundance of flora and fauna.
Levels should have rich ambient soundscapes.
Levels should emphasize verticality in traversal and combat (as in Dishonored 2).
Levels should feature strong atmospheres, such that simply being in them is a source of engagement.
Levels should feature player-driven quiet time.
Levels should use lighting, color, animation, and other techniques to guide the player.
Levels should avoid invisible walls and barriers that the player character could conceivably circumvent.
Levels should utilize visual effects like rain, snow, fireflies, fog, and dust.
Environment art should emphasize animation (i.e., using cable actors and wind).
Transitions between natural environments should feel realistic.
Levels should feel thematically and visually distinct from each other.
Safe areas should feature safe area music (as in Resident Evil 4 or Resident Evil 7).

Game atmosphere references
Deus Ex: Human Revolution
SOMA
Alien: Isolation
S.T.A.L.K.E.R.: Shadow of Chernobyl
Alan Wake
Dark Souls
Resident Evil 1, 4, and 7
Metro
Condemned: Criminal Origins
F.E.A.R.
Silent Hill
Splinter Cell: Chaos Theory

Feedback

Camera feedback
When the PlayerCharacter is high above the ground, the camera will roll left and right, and audio feedback will be played.
When the PlayerCharacter lands after falling a short distance, a sound will play and the camera will translate down and back up again to simulate one's knees bending (as in Shadow Warrior 2013). This translation is faster on the way down, and slower on the way back up.
When the PlayerCharacter lands after falling a greater distance, a louder sound will play, the camera will translate down and back up again, and the camera will pitch down and back up again (as in Dishonored). These effects are faster on the way down, and slower on the way back up. Control is taken away from the player during this sequence.
Moving the camera to the left or right causes the player's weapon to sway slightly in the direction that the camera is turning. This will help convey a sense of speed and efficacy.

Screen space feedback

A vignette appears on the screen when the PlayerCharacter is crouched (as in Dishonored).
A directional indicator is displayed on the screen when the PlayerCharacter takes damage.
An indicator is displayed when the PlayerCharacter is near a live grenade.
An indicator is displayed when the PlayerCharacter's current weapon is out of ammo and they cannot automatically switch weapons.

World space feedback
An objective locator guides the player to their next objective. The locator includes their displacement in meters from the objective.
Hit confirmation is displayed when the PlayerCharacter damages an Enemy. The confirmation can be displayed on the crosshair (as in Rainbow Six: Siege) or on the bone that was hit (as in Deus Ex: Mankind Divided). Hit confirmation is not displayed when the PlayerCharacter damages InteractiveObjects.

Miscellaneous feedback
Slow motion can be used sparingly, such as when the player performs a particularly impressive kill.