Q Director Documentation

For Programmers

プログラマー向けガイド

▼ 日本語版はこのドキュメントの後半にあります（下へスクロール）。 The Japanese edition follows in the second half, further below.

This is a technical reference for programmers who call and integrate Q Director: requirements, the runtime API, components, name resolution, event handling, and lifecycle. The idea: receive motion as a ScriptableObject and just call it by name.

1. For programmers: the call API

Everything is one line. When designers change the motion, no code changes needed.

- Play an animation:
  QDirectorPlayer.Play("AnimName");
- Play / stop a sequence:
  QSequencePlayer.Play("SeqName");
  QSequencePlayer.Stop("SeqName");
- Interaction (runs on click etc., no code):
  Add a QInteractionRunner to the target GameObject and assign the interaction asset built in the Node tab (the Validate tab has a one-click "add" button).
- BGM (Music):
  QDirectorMusic.Play("Title"); // play a playlist by name
  QDirectorMusic.Stop();
  QDirectorMusic.Next(); // skip to the next track
  QDirectorMusic.SetVolume(0.5f); // master volume
  QDirectorMusic.Pause(); QDirectorMusic.Resume();

2. Requirements & assumptions

- **Unity**: Unity 6 (developed & verified on the 6000.3 line).
- **Input**: assumes the New Input System (ENABLE_INPUT_SYSTEM); legacy Input is unused. Click interactions go through the EventSystem, so the scene needs an EventSystem + InputSystemUIInputModule (the Validate tab helps create them).
- **UI**: targets uGUI (RectTransform / CanvasGroup / Graphic family).
- Runs on Unity's main thread (coroutine-based).

3. Call API (runtime, static)

All calls run on the main thread. yield return the returned Coroutine to wait for completion.

```
[Animation: QDirectorPlayer]
  // Play by name (looks up Resources/Animations/<name>.asset)
  Coroutine Play(string animationName, GameObject target = null,
```

```
            Action onComplete = null, int loopCount = 1, float speed = 1f);

  // Target by GameObject name (looked up with GameObject.Find)
  Coroutine Play(string animationName, string targetName,
            Action onComplete = null, int loopCount = 1, float speed = 1f);

  // Direct asset (works outside Resources too)
  Coroutine Play(QAnimationAsset asset, GameObject target = null,
            Action onComplete = null, int loopCount = 1, float speed = 1f);

  // Reverse playback (e.g. a toggle's "close")
  Coroutine PlayReverse(string animationName, GameObject target = null,
                Action onComplete = null, float speed = 1f);

  void Stop(string animationName);   // stop by name identifier
  void StopAll();

  Examples:
    QDirectorPlayer.Play("FadeIn", "TitleCanvas");
    yield return QDirectorPlayer.Play("SlideUp", "TitleLogo"); // wait for completion
    QDirectorPlayer.Play("Pulse", target, loopCount: 0);       // 0 = treated as infinite
```

- When the asset's isLooping = true, runtime loops forever and takes priority over loopCount.

```
[Sequence: QSequencePlayer]
  Coroutine Play(string sequenceName, Action onComplete = null,
            int loopCount = 1, float speed = 1f);
  Coroutine Play(QSequenceAsset asset, Action onComplete = null,
            int loopCount = 1, float speed = 1f);
  // Pre-resolved layer targets (used internally by the component)
  Coroutine Play(QSequenceAsset asset, GameObject[] layerTargets,
            Action onComplete = null, int loopCount = 1, float speed = 1f);
  void Stop(string sequenceName);
  void StopAll();

  Examples:
    QSequencePlayer.Play("IntroSequence");
    QSequencePlayer.Stop("IntroSequence");

[BGM: QDirectorMusic (singleton)]
  static void  Play(string key);     // play the same-named playlist in QMusic.asset
  static void  Stop();
  static void  Next();               // next track within the current playlist
  static void  Pause();  static void Resume();
  static void  SetVolume(float v);   // master volume 0..1 (multiplies each playlist volume)
  static float GetVolume();
  static bool  IsPlaying { get; }
  static string CurrentPlaylist { get; }

  Examples:
    QDirectorMusic.Play("Title");
    QDirectorMusic.SetVolume(0.5f);
```

- Calling Play again with a playlist already playing keeps it going (no re-trigger).
- When autoPlayByScene is on, the scene owns the BGM: on every active-scene change it switches to the playlist assigned to that scene (QMusicAsset.FindForScene: explicit scenes assignment > key match), or stops if none is assigned (no code). Switching to the same playlist is a no-op (seamless). Switch / stop are centralized in QDirectorMusic.BeginEntry /

StopInternal (the extension point for future fades).

4. Runtime components (MonoBehaviour)

- QInteractionRunner
  The engine that runs presentations on triggers like clicks.
- **Fields**: interactionAsset (QInteractionAsset), targetBindings (key->GO).
- Implements IPointerClick/Enter/Exit/Down/Up. Fires via the EventSystem.
- Resolves the key written in a node's "target" to a real GameObject via targetBindings (falls back to name / relative-path search when unbound).
- QAnimationPlayer
  A designer-facing component to play an animation from the Inspector.
- QSequencePlayerComponent
  Plays a sequence from the Inspector. Resolves each layer's targetPath via targetBindings and calls QSequencePlayer.Play(asset, layerTargets, ...).
- QDirectorMusic
  The BGM singleton. Auto-created BeforeSceneLoad, DontDestroyOnLoad. Holds one [QDirectorMusic] GameObject + AudioSource.
- QDirectorFlow
  Manages tutorial / presentation flow (steps + Trigger("key")).
- QAnimEventListener
  A MonoBehaviour that receives [AnimEvent]s from animations / sequences.
- QDirectorSimpleTrigger
  A lightweight component for simple triggers.

5. Name resolution & asset loading

- Play(name) searches with Resources.Load in this order:
  Resources/Animations/<name> (sequences: Sequences/<name>)
  Resources/<name> (fallback to the legacy path)
- BGM loads the single Resources/Music/QMusic.asset.
- The "call name" is animationName / sequenceName inside the asset; for BGM it is QMusicEntry.key.
- When not found, it logs a warning and is a no-op (never throws).

6. Target resolution (important)

- A ScriptableObject (asset) cannot reference a scene GameObject directly.
- So a sequence / interaction "target" is resolved by:
  (1) the component (QSequencePlayerComponent / QInteractionRunner) via targetBindings (key->real object) or by name / relative path.
- The static QSequencePlayer.Play(name) itself does not resolve targetPath.
  When target resolution is needed, resolve the targets and call Play(asset, layerTargets, ...), or play through the component.

- Saved assets lose scene references, so prefer targetBindings (keyed) when you need a reliable binding.

7. Receiving events & cues

When an "event" (lightning) placed on an animation / sequence timeline is reached during playback, the static event QDirectorPlayer.OnAnimEvent fires.

```
public static event Action<string, string> OnAnimEvent;  // (name, eventName)
// name      = animation / sequence name
// eventName = the event name set on the timeline (e.g. "RewardsComplete")
```

There are two ways to receive it.

[Programmer: subscribe in code]

```
void OnEnable()  { QDirectorPlayer.OnAnimEvent += HandleEvent; }
void OnDisable() { QDirectorPlayer.OnAnimEvent -= HandleEvent; }
```

```
void HandleEvent(string name, string eventName)
{
if (eventName == "RewardsComplete") ShowTapToContinue();
}
```

- Subscribe in OnEnable / unsubscribe in OnDisable to avoid leaks. Filter by name to tell which animation / sequence the event came from.

[Designer: no code (QAnimEventListener)]

1. Add QAnimEventListener to the GameObject that should receive the event.
2. Add one entry to Bindings and configure it:

- eventName ⋯ the timeline event name (e.g. "RewardsComplete")
- animationName ⋯ empty = react to that event from any animation / set it to limit to one
- onFire (UnityEvent) ⋯ assign the method to call in the Inspector -> onFire is invoked when that event fires (just like a Button's OnClick). (Internally it subscribes / unsubscribes OnAnimEvent in OnEnable / OnDisable.)

SFX (SoundCue) / VFX (EffectCue) are cues held by the animation and fire automatically at play / editor-preview time (no code).

8. Lifecycle & threading

- Playback is coroutine-based and called from Unity's main thread.
- QDirectorMusic is a singleton + DontDestroyOnLoad, surviving across scenes.
- QDirectorPlayer / QSequencePlayer create their host (runner) internally via EnsureInstance.
- Sequence audio creates a temporary AudioSource per play and disposes it via CleanupRunAudio at loop end / completion / stop. playToEnd (non-loop) is fire-and-forget: it auto-disposes after its length and is not stopped by Stop.

9. Extension & integration

- **Minimal integration**: receive the presentation as a ScriptableObject and just call Play("name"). No code change when designers swap assets.

- **Loosely coupled by name**: callable by string name with no reference, so it is resilient to scene restructuring.
- **Wait for completion**: yield return the returned Coroutine to chain presentations in order.
- **Callbacks**: onComplete lets you hook in work after a presentation finishes.

10. Package structure (included folders)

```
QDirector/              imported to Assets/QDirector/
  Tool/Runtime/   … runtime (playback engine + components). Included in builds.
  Tool/Editor/    … editor extension (this Q Director window). Not in builds.
  Resources/      … the "call by name" asset folders
                  (Animations / Sequences / Interactions / Music/QMusic.asset).
  Samples/        … demo scenes (finished scenes).
  Docs/           … these documents (PDF).
  README.txt      … entry point (getting started, JP/EN).
```

日本語版 / Japanese

本書は、Q Director を呼び出し・統合するプログラマー向けに、動作要件・ランタイム API・コンポーネント・名前解決・イベント受信・ライフサイクルをまとめた技術リファレンスです。基本思想は「演出は ScriptableObject で受け取り、名前で呼び出すだけ」。

1. プログラマー向け：呼び出し API

すべて 1 行で呼べます。デザイナーが演出を変更しても、コードの修正は不要です。

- アニメーション再生：
  QDirectorPlayer.Play("AnimName");
- シーケンス再生 / 停止：
  QSequencePlayer.Play("SeqName");
  QSequencePlayer.Stop("SeqName");
- インタラクション（クリック等で再生・コード不要）：
  対象 GameObject に QInteractionRunner を付け、ノードで作った
  インタラクションアセットを割り当てる（「検証」タブのボタンから付与可能）。
- BGM（音声管理）：
  QDirectorMusic.Play("Title"); // プレイリスト名で再生
  QDirectorMusic.Stop();
  QDirectorMusic.Next(); // 次の曲へ
  QDirectorMusic.SetVolume(0.5f); // マスター音量
  QDirectorMusic.Pause(); QDirectorMusic.Resume();

2. 動作要件・前提

- **Unity**：Unity 6（6000.3 系で開発・検証）。
- **入力**：New Input System を前提（ENABLE_INPUT_SYSTEM）。レガシー Input は不使用。
  クリック等のインタラクションは EventSystem 経由なので、シーンに EventSystem + InputSystemUIInputModule が必要です（検証タブが不足時に作成支援）。
- **UI**：uGUI（RectTransform / CanvasGroup / Graphic 系）を対象とします。
- 実行は Unity メインスレッド前提（コルーチンベース）。

3. 呼び出し API（ランタイム・静的）

すべてメインスレッドから呼びます。戻り値の Coroutine を yield return すれば完了待ちできます。

【アニメーション：QDirectorPlayer】

```
// 名前で再生（Resources/Animations/<name>.asset を検索）
Coroutine Play(string animationName, GameObject target = null,
        Action onComplete = null, int loopCount = 1, float speed = 1f);

// 対象を GameObject 名で指定（GameObject.Find で検索）
Coroutine Play(string animationName, string targetName,
        Action onComplete = null, int loopCount = 1, float speed = 1f);

// アセット直接指定（Resources 外でも可）
```

```
Coroutine Play(QAnimationAsset asset, GameObject target = null,
            Action onComplete = null, int loopCount = 1, float speed = 1f);

// 逆再生（トグルの「閉じる」等）
Coroutine PlayReverse(string animationName, GameObject target = null,
                   Action onComplete = null, float speed = 1f);

void Stop(string animationName);  // 名前を識別子に停止
void StopAll();
```

例：

```
QDirectorPlayer.Play("FadeIn", "TitleCanvas");
yield return QDirectorPlayer.Play("SlideUp", "TitleLogo"); // 完了待ち
QDirectorPlayer.Play("Pulse", target, loopCount: 0);      // 0 = 無限ループ扱い
```

※ アセット側で isLooping = true の場合、ランタイムは常時ループし、loopCount より優先されます。

【シーケンス：QSequencePlayer】

```
Coroutine Play(string sequenceName, Action onComplete = null,
            int loopCount = 1, float speed = 1f);
Coroutine Play(QSequenceAsset asset, Action onComplete = null,
            int loopCount = 1, float speed = 1f);
// レイヤー対象を事前解決して渡す版（コンポーネントが内部利用）
Coroutine Play(QSequenceAsset asset, GameObject[] layerTargets,
            Action onComplete = null, int loopCount = 1, float speed = 1f);
void Stop(string sequenceName);
void StopAll();
```

例：

```
QSequencePlayer.Play("IntroSequence");
QSequencePlayer.Stop("IntroSequence");
```

【BGM：QDirectorMusic（シングルトン）】

```
static void  Play(string key);    // QMusic.asset の同名プレイリストを再生
static void  Stop();
static void  Next();              // 現在のプレイリスト内で次の曲へ
static void  Pause();  static void Resume();
static void  SetVolume(float v);  // マスター音量 0..1（各プレイリスト volume に乗算）
static float GetVolume();
static bool  IsPlaying { get; }
static string CurrentPlaylist { get; }
```

例：

```
QDirectorMusic.Play("Title");
QDirectorMusic.SetVolume(0.5f);
```

※ 同名プレイリストを再生中に再度 Play すると、重ねず継続します（多重再生防止）。※ autoPlayByScene が ON のときは「シーンが BGM を所有」：アクティブシーンの変更ごとに、そのシーンへ割り当てたプレイリスト（QMusicAsset.FindForScene： scenes 明示割当 ＞ key 一致）へ自動で切替／割当が無ければ停止する（コード不要）。同じプレイリストへの切替は何もしない（シームレス継続）。切替・停止は QDirectorMusic.BeginEntry / StopInternal に一元化（将来フェード対応の拡張点）。

4. ランタイムコンポーネント（MonoBehaviour）

- QInteractionRunner
  クリック等のトリガーで演出を起動する実行エンジン。
- **フィールド**：interactionAsset（QInteractionAsset）、targetBindings（key→GO）。
- IPointerClick/Enter/Exit/Down/Up を実装。EventSystem 経由で発火。

- ノードの「対象」に書いたキーを、targetBindings で実 GameObject に解決
  （バインドが無ければ名前／相対パス検索にフォールバック）。
- QAnimationPlayer
  Inspector からアニメを再生するデザイナー向けコンポーネント。
- QSequencePlayerComponent
  Inspector からシーケンスを再生。各レイヤーの targetPath を targetBindings で
  解決して QSequencePlayer.Play(asset, layerTargets, …) を呼ぶ。
- QDirectorMusic
  BGM 管理のシングルトン。BeforeSceneLoad で自動生成、DontDestroyOnLoad。
  [QDirectorMusic] GameObject + AudioSource を 1 つ持つ。
- QDirectorFlow
  チュートリアル・演出フローの管理（ステップ + Trigger("key")）。
- QAnimEventListener
  アニメ／シーケンスの [AnimEvent] を受け取る MonoBehaviour。
- QDirectorSimpleTrigger
  単純なトリガー用の軽量コンポーネント。

5. 名前解決・アセットロードの仕様

- Play(name) は Resources.Load で以下の順に探します：
  Resources/Animations/<name> （シーケンスは Sequences/<name>）
  Resources/<name> （旧パスへのフォールバック）
- BGM は Resources/Music/QMusic.asset を 1 つロードします。
- 「呼び出し名」はアセット内の animationName / sequenceName。BGM は QMusicEntry.key。
- 見つからない場合は警告ログを出して no-op（例外は投げない）。

6. ターゲット解決（重要）

- ScriptableObject（アセット）はシーン上の GameObject を直接参照できません。
- そのため、シーケンス／インタラクションの「対象」は、
  (1) コンポーネント（QSequencePlayerComponent / QInteractionRunner）が
  targetBindings（key→実オブジェクト）や名前／相対パスで解決する、
  という流れになります。
- 静的 API の QSequencePlayer.Play(name) 自体は targetPath を解決しません。
  対象解決が必要な場合は、対象を解決してから
  Play(asset, layerTargets, …) を呼ぶか、コンポーネント経由で再生してください。
- 保存済みアセットはシーン参照を失うため、確実に結びつけたい場合は
  targetBindings（キー指定）の利用を推奨します。

7. イベント・キューの受信

アニメ／シーケンスのタイムライン上に置いた「イベント（☒）」が再生中にその時間へ達すると、静的イベント QDirectorPlayer.OnAnimEvent が発火します。 public static event Action<string, string> OnAnimEvent; // (name, eventName)// name = アニメ名 / シーケンス名// eventName = タイムラインで付けたイベント名（例 "RewardsComplete"）受け取り方は2通り。

【プログラマー：コードで購読】

```
void OnEnable() { QDirectorPlayer.OnAnimEvent += HandleEvent; }
void OnDisable() { QDirectorPlayer.OnAnimEvent -= HandleEvent; }

void HandleEvent(string name, string eventName)
{
   if (eventName == "RewardsComplete") ShowTapToContinue();
}
```

※ OnEnable で購読・OnDisable で解除するとリークしない。
name で「どのアニメ/シーケンスのイベントか」を絞り込める。

【デザイナー：コード不要（QAnimEventListener）】

1. イベントを受けたい GameObject に QAnimEventListener を付ける。
2. Bindings に1件追加して設定する：

- eventName … タイムラインのイベント名（例 "RewardsComplete"）
- animationName … 空欄＝どのアニメの同名イベントにも反応／指定で限定
- onFire（UnityEvent）… 呼びたい関数を Inspector で割り当て
  → 該当イベント発火時に onFire が呼ばれる（Button の OnClick と同じ感覚）。
  （内部では OnEnable/OnDisable で OnAnimEvent を購読／解除している）

効果音（SoundCue）／エフェクト（EffectCue）はキューとしてアニメに含まれ、再生時・エディタプレビュー時に自動発火する（コード不要）。

8. ライフサイクル・スレッド

- 再生はコルーチンベースで、Unity メインスレッドから呼びます。
- QDirectorMusic はシングルトン＋DontDestroyOnLoad でシーンを跨いで存続。
- QDirectorPlayer / QSequencePlayer は内部でホスト（ランナー）を EnsureInstance で生成します。
- シーケンス音声は再生ごとに一時 AudioSource を作り、ループ終端／完了／停止時に CleanupRunAudio で破棄します。playToEnd（非ループ）は撃ちっ放しで、尺ぶん再生後に自動破棄され、停止では止まりません。

9. 拡張・統合のポイント

- **最小統合**：演出は ScriptableObject で受け取り、Play("名前") を呼ぶだけ。
  デザイナーがアセットを差し替えてもコード修正は不要。
- **名前で疎結合**：参照を持たず文字列名で呼べるため、シーン構成変更に強い。
- **完了待ち**：戻り値 Coroutine を yield return して順次演出をつなげられる。
- **コールバック**：onComplete で演出完了後の処理を差し込める。

10. パッケージ構成（同梱フォルダ）

```
QDirector/            インポート先（Assets/QDirector/）
  Tool/Runtime/   … ランタイム（再生エンジン・コンポーネント）。ビルドに含まれる。
  Tool/Editor/    … エディタ拡張（この Q Director ウィンドウ）。ビルドには含まれない。
  Resources/      …「名前で呼ぶ」アセット置き場
                 （Animations / Sequences / Interactions / Music/QMusic.asset）。
  Samples/        … デモシーン（完成シーン）。
  Docs/           … このドキュメント類（PDF）。
  README.txt      … 入口（はじめかた・日英併記）。
```