ゲーム保存のサンプル

*このサンプルは、Microsoft ゲーム開発キットのプレビュー (2019 年 11 月) に対応しています。*

# 説明

このサンプルでは、ゲーム保存データの保存および読み込みに Game Core **XGameSave** API を使用する方法を示します。クラウド同期を処理するための通常の "完全同期" 方式と"オンデマンド同期" モードの両方の使用方法を説明します。前者では、ユーザーが別のコンソールでタイトルをプレイするときに、そのユーザーのすべてのゲーム保存コンテナーが同期されます。後者では、コンテナーを同期するタイミングとその方法をより細かく制御できます。後者は通常、同期に時間がかかる可能性がある大規模なコンテナーでの使用を意図しています。

また、このサンプルでは、新しい Game Core API の表面的な領域に関連するその他の各種手法についても説明します。この関連ファイルのほとんどは、\Helpers\ フォルダーにあり、C++ クラス内の非同期操作のラップ、RAII ラッパー、シングル プレイヤーのユーザー管理、コントローラーのバインドなどの手法を示します。

このサンプルは、単純なシングル プレイヤーのワード パズル ゲーム形式になっています。コントローラーを使用してボードでスポットを選択し、それらの場所に配置する文字を選びます。隣り合う文字は連結されます。有効な英単語を作成すると、文字の頻度に基づいて単語のスコアが計算されます。

このサンプルでは、次のゲーム保存シナリオを扱います。

**完全同期またはオンデマンド同期のモードの使用**

サンプルを起動するときに、そのサンプルで full-sync API

(コンソールとタイトル ストレージ サービスの間ですべてのゲーム保存データが同期される) と sync-on-demand API (必要時のみゲーム保存データが同期される) のどちらを使用するかを選択できます。実際の典型的なタイトルでは、ゲームのニーズに基づいてこの決定がハード コーディングされます。

現在出荷されているタイトルのうち、オンデマンド同期モードを使用しているものはほとんどありません。すべてのゲーム データを同期するような、大量**かつ**大規模なコンテナーを使用する傾向があるものは、ユーザーが同期の完了を待たなければならず、不便です。

注:既に完全同期を実行していて、オンデマンド同期を試してみたい場合は、別のユーザーでサインインするか、ゲーム保存データのローカル キャッシュをクリアする必要があります。

Xbox One のローカル キャッシュをクリアするには、XDK コマンド プロンプトから "xbstorage.exe reset /force" コマンドを実行します。

Windows 10 PC のローカル キャッシュをクリアするには、管理者コマンド プロンプトから "gamesaveutil.exe reset" コマンドを実行します。

注: Windows May 2019 SDK をインストールした場合、**gamesaveutil.exe** は次のディレクトリにあります。  
%ProgramFiles(x86)%\Windows Kits\10\Extension SDKs\XboxLive\1.0\Bin\x64

**ゲーム保存データの読み込み、保存、削除を行う**

メニュー オプションを使用して、ゲーム ボードの読み込み、保存、削除を行います。最大 9 つの異なるボードを保存できます。

**コンテナーと BLOB を一覧表示する**

メニュー オプションを使用して、コンテナーと BLOB を列挙します。出力は、ゲーム画面のスクロール可能なデバッグ出力領域に表示されます。

**最終更新日と残りのクォータを表示する**

この情報は、ゲーム画面のタイトルのすぐ下に表示されます。

**ユーザーのサインアウト時に自動保存する**

現在のゲーム ボードがまだ保存されていない場合は、ユーザーがサインアウトしたときに自動的に保存されます。

**中断時の自動保存**

中断しているイベントに対して、現在のゲーム ボードがまだ保存されていない場合は、自動的に保存されます。

# サンプルのビルド

Xbox One の devkit を使用している場合は、アクティブなソリューション プラットフォームを Gaming.Xbox.XboxOne.x64 に設定します。

Project Scarlett を使用している場合は、アクティブなソリューション プラットフォームを Gaming.Xbox.Scarlett.x64 に設定します。

*詳細については、GDK ドキュメントの* 「サンプルの実行」*を参照してください。*

# サンプルの使用

**Xbox Live サンドボックスの要件**

* Xbox One devkit: コンソールのサンドボックスを XDKS.1 に設定します

**起動メニュー**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  | | --- | --- | --- | | 操作 | コントローラー | キーボード | | "完全同期" モードと "オンデマンド同期" モードの選択を切り替える | 左スティックまたは方向パッド | 方向キー | | メニュー項目を選択する | A ボタン | Enter | |

**ゲーム ボード**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  | | --- | --- | --- | | 操作 | コントローラー | キーボード | | カーソルを移動する | LS または方向パッド | 方向キー | | メニュー項目を選択する | A ボタン | Enter | | ゲーム保存スロットを選択する | LB/RB ボタン | 1 から 9 のキー | | カーソルの下の文字のタイルを変更する | RS の左/右 | A から Z のキー | | カーソルの下の文字をクリアする | X ボタン | Delete または Space | | デバッグ出力をスクロールする | RS の上/下 | PageUp/PageDn  および  Home/End | |

# ゲーム メニューに関する注意事項

* **ボードの読み取り**この作業の大部分は、**GameSaveManager::ReadBlocking** で実行されます。 **XGameSaveCreateContainer** および **XGameSaveReadBlobData** APIを使用して、現在のゲーム保存スロットのゲーム ボードが読み込まれます。
* **ボードの保存**この作業の大部分は、**GameSaveManager::SaveBlocking** から始まります。**XGameSaveCreateContainer、XGameSaveCreateUpdate、XGameSaveSubmitBlobWrite、XGameSaveSubmitBlobDelete** および **XGameSaveSubmitUpdate** API を使用して**、**現在のゲーム保存スロットのゲーム ボードが保存されます。
* **ボードのリセット**  
  ボードからすべての文字をクリアします。ボードが以前に保存されている場合、ボードは"ダーティ" とマークされます。ボードがまだ保存されていない場合、ボードは "ダーティ" とマークされません。
* **ボードの削除**この作業は **GameSaveManager::DeleteBlocking** の内部で実行されます。**XGameSaveDeleteContainer** API を使用して、現在のゲーム保存スロットのゲーム ボードが削除されます。
* **ボードの BLOB の削除**作業の大部分が **GameSaveManager::DeleteBlobsBlocking** によって処理されます。**XGameSaveCreateContainer、XGameSaveCreateUpdate、XGameSaveSubmitBlobDelete、**および **XGameSaveSubmitUpdate** API を使用して**、**現在のゲーム ボードの BLOB のみが削除され、空のコンテナーが残ります。
* **コンテナーの一覧表示**この作業は **GameSaveManager::EnumerateContainersBlocking** で実行されます。**XGameSaveEnumerateContainerInfo** API を使用してすべてのコンテナーが列挙され、ゲーム画面のデバッグ出力領域に一覧表示されます。
* **コンテナーと BLOB の一覧表示**この作業は **GameSaveManager::EnumerateContainersBlocking** で実行されます。**XGameSaveEnumerateContainerInfo、XGameSaveCreateContainer** および **XGameSaveEnumerateBlobInfo** API を使用してすべてのコンテナーと BLOB が列挙され、ゲーム画面のデバッグ出力領域に一覧表示されます。

ゲームのプレイに関する注意事項

**ゲームのプレイ**

ゲームは 5 x 5 のグリッドでプレイされます。グリッドの任意の場所に文字を配置できます。ボードの横方向または下方向の連続文字が一般に認められた英単語を形成している場合、単語の各文字のポイント値の合計に基づいて得点が加算されます。目標は、スコアを最大にすることです。各ボードに置く各文字数には制限があります。残りの文字数は、ゲーム ボードのすぐ上で追跡されます。

**ゲーム ボードの読み込み**

デモの目的のため、ゲーム ボードの画面が表示されたときや、新しいゲーム保存スロットに切り替えたときに、ゲーム ボードは**自動的に読み込まれません**。これにより、特定のゲーム保存スロットにおいて、読み込みおよび保存の操作をユーザーが完全に制御できます。

**変更されたゲーム ボードと自動保存**

ゲーム ボードで文字が変更されたとき、またはユーザーが [リセット] メニュー コマンドを使用したとき、ゲーム ボードは "ダーティ" とマークされます (画面の上部にあるボード名の後のアスタリスクで示されます)。ダーティなゲーム ボードは、次の条件で自動保存されます。

* 別のゲーム ボードに切り替える (コントローラーの LB/RB)
* ユーザーのサインアウト
* ゲームの中断

# 実装に関する注意事項

**GameSaveManager** クラスは、ゲームのゲーム保存操作を管理します。**InitializeForUser()** メソッドにより、プレイヤーの Connected Storage 保存コンテキストが設定されます。保存データの読み込み、保存、列挙、削除のメソッドもあります。クラスの各メソッドの使用上の注意事項については、ヘッダー ファイルのコメントをご覧ください。

このサンプルには、ユーザー管理およびコントローラー管理の機能がいくつかありますが、*これらの機能を説明することは意図されていません*。 このサンプルは、さまざまな状況でのゲーム保存の使用方法について理解することを目的としています。

ゲームで使用されるゲーム データ構造には、インデックスとゲーム ボードの 2 種類があります。テンプレート化された **GameSave** クラスには、あらゆる種類のゲーム データ用にデータの読み込みと保存を行うために **GameSaveManager** で使用する汎用的なメソッドが用意されています。GameSaveManager.h の **GameBoardIndex** 構造体で定義されているインデックスは主に、プレイヤーが使用した最終保存スロット ("アクティブなボード") を追跡するために使用されます。ゲーム ボードのデータは、GameBoard.h の **GameBoard** 構造体で表されます。

# その他のサンプル コード

このサンプルには、Game Core API の操作をサポートするために、独自のコードで使用する可能性がある一連の付加的なヘルパー クラスが付属しています。これらのファイルのほとんどは、\Helpers\ フォルダーにあります。

|  |  |  |
| --- | --- | --- |
| フォルダー | ファイル名 | 説明 |
| \Helpers\ | HandleWrapperBase.h  XGameSaveHandleWrappers.h XTaskQueueHandleWrapper.h XTaskQueueHandleWrapper.cpp XUserHandleWrapper.h | サンプルで使用する主要な API で定義された Game Core ハンドル タイプをラップする RAII クラス ラッパー。 |
| \Helpers | User.h  User.cpp | ユーザーのタイトルで保持でき、クエリを実行できるユーザー オブジェクト。その本質は、XUserHandle です。これは、シングル サインインしたユーザー マネージャーによって返されるオブジェクトの種類です。 |
| \Helpers | UserManager.h  UserManager.cpp | シングル サインイン ユーザー マネージャー。ゲームをプレイしている 1 人のユーザーを追跡し、別のユーザーを選択できるようにします。 |
| \Helpers | UTF8Helper.h  UTF8Helper.cpp | UTF-8 のテキスト変換の関数。 |
| \Helpers | Buffer.h  Buffer.cpp | BLOB 処理のクラス。メモリを割り当て、割り当てられたバッファーの開始位置とサイズを追跡し、オブジェクトが破棄されたときにメモリを解放します。移動セマンティクスもサポートされます。 |
| \Helpers | StateMachine.h | 状態機械のロック フリー実装。(ロック フリーで、かつ状態機械が特定の状態に変わるまでコンシューマーが待機できるバージョンも提供されています)。 |
| \Helpers | AsyncOp.h  AsyncOp.cpp | API の非同期呼び出しの進捗状況を追跡するために拡張できる基底クラス。AsyncTask 実装も提供され、非同期タスク キューに対して操作を実行できます。 |
| \Helpers | AsyncAction.h | 非同期タスク キューで std::function またはラムダを実行し、生成された結果 (存在する場合) をユーザーが取得できるテンプレート ベースのクラス。これは、ppltasks と同様に機能します。 |
| \Helpers | ScopedLockWrappers.h | Slim Reader/Writer ロックをラップする範囲付きラッパー。試行入力や、ロックの範囲内のペイロード データへのアクセスのみの提供などのサポートが含まれます。 |
| \Helpers | TaskQueue.h TaskQueue.cpp | 初期化時、コアごとに 1 つのワーカー タスク ディスパッチ キューと 1 つの作業完了キューを作成します。ユーザーは、そのコアのタスク キューに対してクエリを実行できます。  **注:** このコードは、起動時に*中断される* スレッドを適切に作成して、そのスレッドを再開する前に特定のコアに親和化する方法を示しています。これにより、既存のコアでの作業の妨害や、コア ホッピングすることなく、スピンアップできます。 |
| \Common\ | InputDeviceManager.h  InputDeviceManager.cpp  ScopedGameInputDeviceInfo.h GamePad.h GamePad.cpp | コントローラーの接続および切断のイベント、ユーザーとデバイスの関連付け、さらに測定している GameInput 読み取りから従来の DirectXTK コントローラーの実装までを追跡するコントローラー マネージャー。 |
| \ | Assets.h Assets.cpp SampleSpecificAssets.inl | 簡単なストリーミング資産管理システムの基盤となるもの。以前のバージョンと比べてサンプルの資産管理を一元化することで、PC での紛失デバイス処理を簡素化するために使用することもできます。 |

## 非同期キューの使用

このサンプルでは、タスクと通知を既定のシステム スレッド プールに送信するタスク キューを作成します。サンプル コードによって作成された非同期ワークロードはすべて、このキューに入ります。これらのタスクは、スレッド プールで次に使用できるスレッドを実行します。

ユーザーとデバイスの関連付け、ユーザーのイベントおよび GameInput 関連のコールバック作業がすべて、Common\InputDeviceManager.h で定義されている **DEFAULT\_INPUT\_WORK\_AND\_CALLBACK\_CORE** のキューに入れらます。目的は、これらのイベントをすべて 1 つのスレッドに強制的にシリアル化することで、より簡単に処理できるようにすることです。

## スレッド セーフ

ユーザー マネージャーのクラスとコントローラー関連のクラスでは、スレッド間の同期に従来のヘビーウェイトのロック機構が使用されるため、スレッド セーフです。ただし、再入可能ではない場合があります。

# 既知の問題

* パワーダウン アイドル状態に入っている **GameInput** コントローラーによって、GameInput ライブラリがクラッシュします。

# 更新履歴

**初期リリース:2019 年 8 月**

新しいバージョン。大部分が新しいタスク システムの Game Core API の使用するために書き換えられています。同期 (ブロッキング) の GameSave 呼び出し、新しい資産ローダー コード、新しいユーザー管理作業を使用します。すべての WinRT 関連コードが削除されています。また、このバージョンでは PC サポートが削除されていますが、今後のリリースに追加されます。

# プライバシーに関する声明

サンプルをコンパイルして実行すると、サンプルの使用状況を追跡するため、サンプルの実行可能ファイルのファイル名が Microsoft に送信されます。このデータ収集を無効にするには、プロジェクトの設定で、C/C++ / プリプロセッサ / プロセッサ定義の一覧から ATG\_ENABLE\_TELEMETRY を削除します。

Microsoft のプライバシーに関する声明の詳細については、「[Microsoft プライバシー ステートメント](https://privacy.microsoft.com/en-us/privacystatement/)」を参照してください。