**WordGame 游戏保存示例 (C++/WinRT)**

此示例要求 2017 年 3 月或之后的 Xbox One XDK。

# 描述

此示例使用连接的存储 API 演示了 Xbox One XDK 的游戏保存功能，以构建完整的端到端游戏体验。它演示了通过 **C++/WinRT** 投影标头使用连接的存储。此示例包括与 C++/WinRT 投影兼容的 **IBuffer** 接口的简单实施，并演示了如何使用 **CoRoutines** 进行并行编程。

此示例为以下游戏保存方案提供选项。

* **使用“完全同步”或“按需同步”模式**

出于演示目的，在示例启动时，可以选择使用完全同步 API（在控制台和标题存储服务之间同步所有游戏保存数据）或按需同步 API（仅按需同步游戏保存数据）。  
注意：如果您已经执行完全同步且想要尝试按需同步，则应该使用其他用户登录，或清除游戏保存数据的本地缓存。

* + 要清除 Xbox One 的本地缓存，可以从 XDK 命令提示符运行“xbstorage.exe reset / force”命令。
* **加载、保存和删除游戏保存数据**

使用菜单选项加载游戏板（使用“获取”或“读取”API），予以保存并删除。最多可以保存 9 个不同的游戏板。

* **列出容器和 blob**

使用菜单选项，以枚举容器和 blob。输出显示于游戏屏幕的可滚动调试输出区域。

* **查看上次修改日期和剩余配额**

此信息显示于游戏屏幕标题的正下方。

* **用户退出时自动保存**

如果当前游戏板尚未保存，则在用户退出时将自动保存。

* **暂停时自动保存**对于暂停事件，如果当前游戏板尚未保存，则将自动保存。

# 构建示例

**XDK 构建要求**

* Visual Studio 2017（15.3 更新）或更高版本
* Xbox One XDK

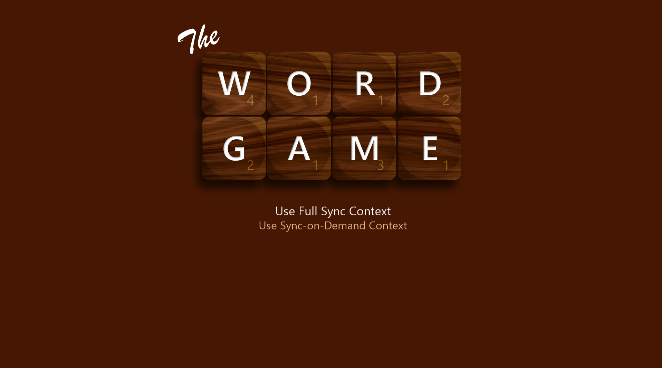
# 使用样本

**Xbox Live 沙盒要求**

* 将开发工具包的沙盒设置为 **XDKS.1**

**启动菜单**

|  |  |
| --- | --- |
| 操作 | 游戏手柄 |
| 在“完全同步”  模式和  “按需同步”  模式之间选择 | LS 或 D-Pad |
| 选择菜单项 | A 键 |



**游戏板**

|  |  |
| --- | --- |
| 操作 | 游戏手柄 |
| 移动光标 | LS 或 D-Pad |
| 选择菜单项 | A 键 |
| 选择游戏保存槽 | LB/RB 按钮 |
| 更改光标下的字母图块 | RS 左/右 |
| 清除光标下的字母 | X 键 |
| 滚动调试输出 | RS 上/下 |



# 游戏菜单备注

* **获取板**使用 **GetAsync** API，为当前游戏保存槽加载游戏板。
* **读取板**  
  使用 **ReadAsync** API（GetAsync 的替代方法），为当前游戏保存槽加载游戏板。
* **保存板**  
  使用 **SubmitUpdatesAsync** API，为当前游戏保存槽保存游戏板。
* **重置板**  
  清除所有字母的板。如果板先前已保存，则会将板标记为“已更新”。如果板尚未保存，则板将不会被标记为“已更新”。
* **删除板**使用 **DeleteContainerAsync** API ，以删除当前游戏保存槽的游戏板。
* **删除板 Blob**  
  使用 **SubmitUpdatesAsync** API，以仅删除当前游戏板的 blob，留下一个空容器。
* **列出容器**  
  使用 **GetContainerInfo2Async** API，以枚举所有容器，并将它们列于游戏屏幕的调试输出区域。
* **列出容器和 Blob**使用 **GetContainerInfo2Async** 和 **GetBlobInfoAsync** API，以枚举所有容器和 blob，并将它们列于游戏屏幕的调试输出区域。

游戏玩法笔记

**游戏玩法**

游戏以 5 x 5 网格进行。可以在网格上的任何位置放置字母。形成公认英语单词的连续字母，无论是沿板横向还是纵向，都将根据单词中每个字母的总分值得分。目标是将分数最大化。每个板上放置的每个字母都有数量限制。剩下的计数可在游戏板正上方进行追踪。

**游戏板加载**

出于演示目的，当游戏板屏幕出现时，或当切换到新游戏保存槽时，游戏板**不要自动加载**。这使开发人员有机会选择加载方法（“获取”或“读取”），并可在特定游戏保存槽完全控制加载和保存操作。

**改变游戏板和自动保存**

当游戏板上的字母有更改时，或者使用“重置”菜单命令时，游戏板将被标记为“已更新”（在屏幕顶部的板名称后面用星号表示）。在以下条件下，将自动保存已更新的游戏板：

* 切换到不同的游戏板（游戏手柄 LB/RB）
* 用户退出
* 游戏暂停

# 实施说明

**GameSaveManager** 类将管理游戏的游戏保存操作。**InitializeForUser()** 方法为玩家设置连接的存储保存上下文。还有加载、保存、枚举和删除保存数据等方法。有关类中每个方法的使用说明，请参阅标头文件中的注释。

游戏使用了两类游戏数据结构：索引和游戏板。模板化 **GameSave** 类提供了 GameSaveManager 用于为任何类型的游戏数据一般加载和保存数据的方法。由 GameSaveManager.h 中 **GameBoardIndex** 结构定义的索引主要用于跟踪玩家上次所用的保存槽（“活动板”）。游戏板数据由 GameBoard.h 中的 **GameBoard** 结构表示。

# 已知的问题

没有。

# 更新历史记录

**初始发行：*2017 年 10 月***

与 2017 年 10 月的 XDK 一同发布

# 隐私声明

编译和运行示例时，示例可执行文件的文件名将发送给Microsoft以帮助跟踪示例使用情况。要选择退出此数据收集，您可以删除Main.cpp中标记为“Sample Usage Telemetry”的代码块。

有关 Microsoft 隐私政策的更多信息，请参阅 [Microsoft 隐私声明](https://privacy.microsoft.com/zh-cn/privacystatement/)。