**WordGame 게임 저장 샘플 (C++/WinRT)**

이 샘플에는 Xbox One XDK (2017 년 3 월 이후)가 필요합니다.

# 설명

이 샘플은 연결된 스토리지 API를 사용한 Xbox One XDK의 게임 저장 기능을 보여주며 완벽한 종단 간 게임 경험을 제공합니다. 그것은 **C++/WinRT** 투영 헤더를 사용한 연결된 스토리지의 사용법을. 보여줍니다. 이 샘플은 C++/WinRT 투영과 호환되는 **IBuffer** 인터페이스의 간단한 구현을 포함하며 병렬 프로그래밍에 대한 **CoRoutines** 사용을 보여줍니다.

이 샘플은 다음 게임 저장 시나리오에 대한 옵션을 제공합니다.

* **"전체 동기화"또는 "주문형 동기화"모드 사용**

데모 목적으로 샘플 실행시 (콘솔과 제목 저장 서비스간에 모든 게임 저장 데이터를 동기화하는) 전체 동기화 API 또는 (필요에 따라 게임 저장 데이터 만 동기화하는) 동기화 요청 API 중 하나를 사용하도록 선택할 수 있습니다.  
알림: 이미 전체 동기화를 수행했으며 요청시 동기화를 시도하려면 다른 사용자로 로그인하거나 게임 저장 데이터의 로컬 캐시를 지워야 합니다.

* + Xbox One의 로컬 캐시를 지우려면 XDK 명령 프롬프트에서 "xbstorage.exe reset/force” 명령을 실행하십시오.
* **게임 저장 데이터 로드, 저장 및 삭제**

메뉴 옵션을 사용하여 (Get 또는 Read API를 사용하여) 게임 보드를 로드하고 저장하고 삭제하십시오. 최대 9개의 다른 보드를 저장할 수 있습니다.

* **목록 컨테이너 및 블로브**

메뉴 옵션을 사용하여 컨테이너와 블로브를 열거하십시오. 출력은 게임 화면의 스크롤 가능한 디버그 출력 영역에 표시됩니다.

* **최종 수정 날짜 및 남은 저장 용량 보기**

이 정보는 게임 화면의 제목 바로 아래에 표시됩니다.

* **사용자 로그 아웃시 자동 저장**

현재 게임 보드가 아직 저장되지 않은 경우 사용자가 로그 아웃하면 자동으로 저장됩니다.

* **일시 중지시 자동 저장**일시 중지 이벤트에 대한 응답으로 현재 게임 보드가 아직 저장되지 않은 경우 자동으로 저장됩니다.

# 샘플 빌드

**XDK 빌드 요구 사항**

* Visual Studio 2017 (15.3 업데이트) 이상
* Xbox One XDK

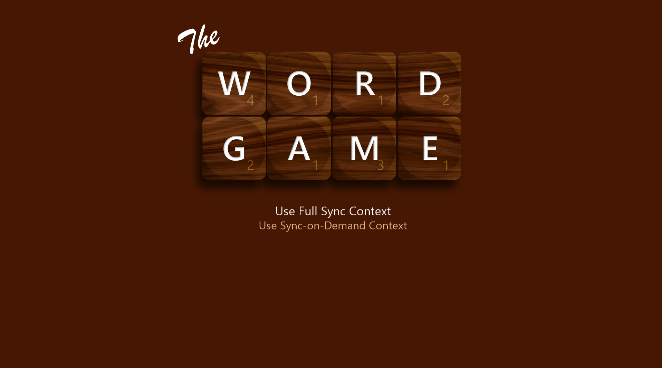
# 샘플 사용하기

**Xbox Live Sandbox 요구 사항**

* 귀하의 devkit의 샌드 박스를 **XDKS.1**로 설정

**시작 메뉴**

|  |  |
| --- | --- |
| 액션 | 게임 패드 |
| '전체 동기화'모드와 '주문형 동기화'모드를 선택하세요 | LS 또는 D 패드 |
| 메뉴 항목 선택 | A 버튼 |



**게임 보드**

|  |  |
| --- | --- |
| 액션 | 게임 패드 |
| 커서 이동 | LS 또는 D 패드 |
| 메뉴 항목 선택 | A 버튼 |
| 게임 저장 슬롯 선택 | LB/RB 버튼 |
| 커서 아래 문자 타일 변경 | RS 좌/우 |
| 커서 아래의 문자 지우기 | X 버튼 |
| 스크롤 디버그 출력 | RS 위/아래 |



# 게임 메뉴 노트

* **보드 가져 오기  
  GetAsync** API를 사용하여 현재 게임 저장 슬롯을 위한 게임 보드를 로드합니다.
* **보드 읽기**  
  **ReadAsync**API(GetAsync의 다른 방법)를 사용하여 현재 게임 저장 슬롯에 게임 보드를 로드합니다.
* **보드 저장**  
  **SubmitUpdatesAsync** API를 사용하여 현재 게임 저장 슬롯을 위한 게임 보드를 저장합니다.
* **보드 재설정**  
  모든 문자 보드를 지웁니다. 보드가 이전에 저장되었다면 보드에 "더티 (dirty)"라고 표시됩니다. 보드가 아직 저장되지 않은 경우, 보드는 "더티 (dirty)"로 표시되지 않습니다.
* **보드 삭제  
  DeleteContainerAsync** API를 사용하여 현재 게임 저장 슬롯에 대한 게임 보드를 삭제합니다.
* **보드 블로브 삭제**  
  **SubmitUpdatesAsync** API를 사용하여 빈 컨테이너를 남겨두고 현재 게임 보드에 대한 블로브를 삭제합니다.
* **목록 컨테이너**  
  **GetContainerInfo2Async** API를 사용하여 모든 컨테이너를 열거하고 게임 화면의 디버그 출력 영역에 나열합니다.
* **목록 컨테이너 및 블로브  
  GetContainerInfo2Async** 과 **GetBlobInfoAsync** API를 사용하여 모든 컨테이너와 blob을 열거하고 게임 화면의 디버그 출력 영역에 나열합니다.

게임 플레이 노트

**게임 플레이**

게임은 5 x 5 그리드로 재생됩니다. 그리드의 아무 곳에나 문자를 배치할 수 있습니다. 보드를 가로 질러 또는 아래에 인식된 영어 단어를 형성하는 연속된 글자는 단어의 각 글자에 대한 점수 값의 합계에 기초하여 점수를 얻습니다. 목표는 점수를 최대화하는 것입니다. 각 문자에는 제한된 수의 글자를 사용해야 합니다. 나머지 카운트는 게임 보드 바로 위에서 추적됩니다.

**게임 보드 로딩**

데모 용으로 게임 보드는 게임 보드 화면이 나타나거나 새로운 게임 저장 슬롯으로 전환 할 때 **자동으로 로드되지 않습니다.** 이렇게 하면 개발자가 특정 게임 저장 슬롯에서로드 메소드 (가져 오기 또는 읽기)를 선택하고 로드 및 저장 조작을 완전히 제어할 수 있습니다.

**게임 보드 변경 및 자동 저장**

게임 보드에서 글자가 변경되거나 메뉴 재설정 명령을 사용하면 게임 보드가 "더티"(화면 상단의 보드 이름 뒤에 별표로 표시)로 표시됩니다. 더러운 게임 보드는 다음 조건에서 자동 저장됩니다:

* 다른 게임 보드 (게임 패드 LB/RB)로 전환
* 사용자 로그 아웃
* 게임 일시 중지

# 구현 정보

**GameSaveManager** 클래스는 게임에 대한 게임 저장 작업을 관리합니다. **InitializeForUser()** 메소드는 플레이어에 대한 연결된 스토리지 저장 컨텍스트를 설정합니다. 저장 데이터를 로드, 저장, 열거 삭제하는 메소드도 있습니다. 클래스의 각 메소드에 대한 사용법은 헤더 파일의 주석을 참조하십시오.

게임에서 사용하는 게임 데이터 구조에는 인덱스와 게임 보드의 두 가지 유형이 있습니다. 템플리트화 된 게임저장 (**GameSave**) 클래스는 GameSaveManager에서 모든 유형의 게임 데이터에 대해 데이터를 로드하고 저장하는 데 사용할 메서드를 제공합니다. 색인은 GameSaveManager.h에서 **GameBoardIndex** struct로 정의되며 주로 플레이어 ( "활성 보드")가 사용한 마지막 저장 슬롯을 추적하는 데 사용됩니다. 게임 보드 데이터는 GameBoard.h 의 게임 보드 (**GameBoard**) struct표시됩니다.

# 알려진 문제점

없음.

# 업데이트 기록

**초판: *2017년 10월***

2017년 10월 XDK와 함께 출시되었습니다.

# 개인정보처리방침

샘플을 컴파일하고 실행할 때 샘플 실행 파일의 이름이 Microsoft로 보내져 샘플 사용을 추적 할 수 있습니다. 이 데이터 수집을 거부하려면 Main.cpp에서 "샘플 사용 텔레메트리"라는 코드 블록을 제거하면 됩니다.

Microsoft의 개인 정보 취급 방침에 대한 일반적인 내용은 [Microsoft 개인 정보 취급 방침](https://privacy.microsoft.com/ko-kr/privacystatement/)을 참조하십시오.