소형 장치, 대형 게임: 대형 공유 스크린의 멀티플레이어 게이밍
(Carry Small, Game Large: Big Shared Screen Multiplayer Gaming)

Omar Rodriguez , Erik J. Johnson , Scott Crabtree and Brad Werth

2008. 5. 6

http://www.gamasutra.com/view/feature/3649/carry_small_game_large_big_.php

포터블 게임의 크기는 작아야 하는가? 여러 플레이어가 동일한 물리적 공간에 있으면서 대형 스크린을 함께 바라보고 있다면 어떻게 될까? 대형 스크린으로 모든 액션을 볼 수 있고 각 플레이어가 랩톱 또는 핸드헬드 장치를 통해 아바타를 관리한다면 어떻게 될까?

이러한 사항을 알아보기로 결정했다. 따라서 새로운 게임플레이 모델을 테스트하기 위해 프로토타입 프레임워크를 구축했다. 이 새로운 모델이 아주 재미있고 기존 게임에 구현하고 추가하기에 어렵지 않다고 생각했다. 또한 프레임워크와 게임 코드를 제공하여 귀하도 시도해 보도록 할 생각이다.

Intel 팀의 일원으로 게임을 철저히 살펴 보면서 소형 장치의 게임은 소형 게임이어야 한다는 견해에 이의를 제기하기로 했다. 그 대신 다음과 같이 가정하는 경우 "소형 장치, 대형 게임" 프레임워크가 어떤 형태일 수 있을까를 알아보기 시작했다.

- 게임 클라이언트는 모바일 인터넷 장치 또는 인터넷에 연결된 소형 폼 팩터 *PC*이다.
- 게임은 멀티플레이어용이다(대부분의 게임이 최상의 상태로 플레이된다고 믿고 있음).
- 게임 프레임워크는 플레이어 집단이 모였다 흩어질 때 이들을 적절하게 다루어야 한다. 이것은 게임 클라이언트에 게임 소프트웨어를 설치하지 않음을 의미한다.
- 모든 플레이어는 일반적인 대형 공유 스크린에 액세스한다.
- 지원되는 게임은 “트위치(*twitch*)” 게임이 아니다. 프레임워크의 반응성이 네트워크 속도에 따라 달라지는 반면 프레임워크 자체는 트위치 게임에 적합하지 않은 지연 기능을 도입한다.

그 결과는 그림 1의 탱크 게임과 멀티플레이어 조각그림 맞추기 등 우리가 생산한 두 가지 게임의 상단에 있는 게임 프레임워크이다.

그림 1 (왼쪽) 소형 폼 팩터 PC에 적합한 클라이언트 스크린 및 (오른쪽) 게임의 대형 공유 스크린

이러한 새로운 게임플레이 모델은 어떤 방식으로 어디서 사용될 것인가?

마지막 가정(공유 스크린)은 파격적인 만큼 흥미를 자아낸다. 게임 참가자는 물리적으로 동일한 장소에 위치해야 하므로 새로운 게임 모델을 사용할 수 있는 장소가 한정된다. 이 게임 모델을 커피숍, 몰, 극장, 컨벤션 등에서 사용하는 것을 상상해 본다. 또한 현재 집단 게임이 플레이되는 바에서도 이 모델이 사용되고 있으며 인터넷 케이블 장치와는 대조적으로 독점 컨트롤러가 갖추어져 있다.

공유 스크린 가정은 제한적이지만 게임 디자이너가 자유롭게 인간 상호간의 무수한 커뮤니케이션을 활용할 수 있다. 사람들이 동일 스크린을 볼 수 있으므로 서로를 볼 수 있고 상호간의 대화를 들을 수 있다. 또한 집단에서 이러한 방식으로 플레이하려고 시도할 때 플레이어들이 동일한 공간에 있으므로 전략을 짜며 서로 웃을 수 있다. 게임 내의 조롱 섞인 말투는 게임 분위기를 활기차게 만드는 역할을 한다. 멀티플레이어 게임의 좋지 않은 면을 억누르기 위해 익명을 허용하지 않는 상황과는 거리가 멀다.

게임 프레임워크

사용 모델에서는 플레이어가 신속하고 쉽게 게임에 참여하고 떠나는 것으로 설정되어 있다. 이러한 목적을 달성하기 위해 팀의 견해에 따라 웹 애플리케이션의 구조를 살펴보았다. 웹 애플리케이션은 웹 브라우저 내부에서 실행되므로 클라이언트에 소프트웨어가 거의 필요없다. 웹 브라우저 내부에서 실행하기 위해 사용 모델을 구현하여 유사 모델을 따르기로 결정했다. 클라이언트에 대한 유일한 요구사항은 현대적인 그래픽의 웹 브라우저를 갖추는 것이다.

게임을 플레이하려면 플레이어는 브라우저를 열고 게임 URL로 이동하여 이름을 입력한다. 그러면 대형 화면에 해당 캐릭터가 나타난다. 플레이어는 고유 이름과 아이콘을 할당받고 입력과 점수를 계속 지켜본다. 또한 휴대용 장치에 있는 브라우저의 컨트롤을 사용하여 아바타를 관리한다. 플레이어의 입력은 JavaScript/AJAX를 통해 플레이어 스크린에서 서버 데이터베이스까지 비동기식으로 전달된다.

서버 측에서는 PHP, JavaScript/AJAX 및 관계형 데이터베이스 등 잘 알려진 웹 기술을 사용한다. 웹 브라우저에서 실행되는 공유 스크린은 데이터베이스에서 발견된 새로운 플레이어의 입력에 의해 업데이트된다. 현재 게임 인스턴스를 추적하기 위해 데이터베이스는 플레이어의 입력에 대한 비동기식 공유 큐로 사용된다. 또한 현재 게임 인스턴스는 데이터베이스에 저장된 고유 ID로 식별하고 플레이어에게 게임의 재시작을 통지하기 위해 사용된다. 저장된 게임 ID가 변경되면 데이터베이스는 업데이트되고 플레이어는 새로운 게임에 자동으로 참여하게 된다.

구조는 게임 클라이언트 및 게임 서버 등 두 가지 주요 구성요소로 이루어져 있다. 게임 클라이언트에는 게임에 대한 플레이어의 인터페이스가 포함되어 있다. 게임 서버는 플레이 필드를 표시하는 UI와 데이터베이스에서 플레이어의 입력을 검색하는 코드로 이루어져 있다. 전체 게임은 생산자와 소비자 문제에 대한 것이다. 플레이어는 생산자이고 게임은 소비자이며 데이터베이스는 공유 버퍼로 역할한다. 우리 팀의 경우, 데이터베이스 시스템은 생산자와 소비자 문제에서 동기화 이슈를 처리한다. 이 프레임워크는 그림 2에 나와 있다.

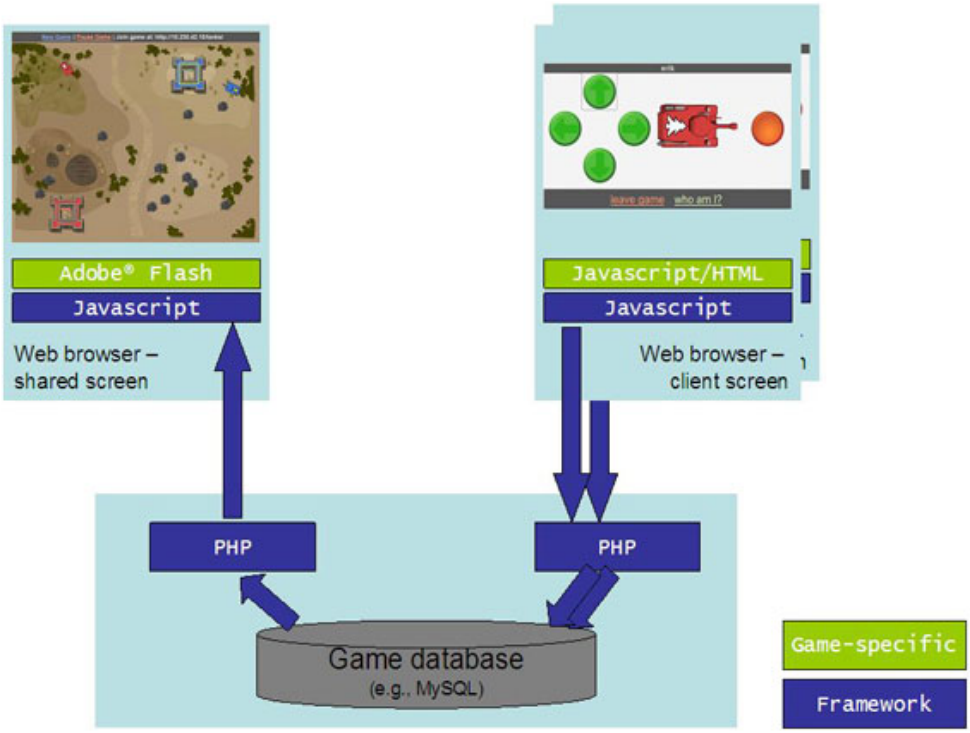

그림 2 게임 프레임워크의 주요 구성요소

게임 클라이언트

게임이 대형 스크린에 표시되지만 플레이어들은 로컬 게임 클라이언트와 상호작용을 수행한다. 게임 클라이언트는 캐릭터 작성에 사용한 폼을 플레이어에게 제공한다. 플레이어가 일단 로그인하면 게임 클라이언트의 브라우저에서 실행하는 JavaScript코드는 키스트로크(keystroke) 또는 스크린에서 발견된 버튼의 클릭 형태로 입력을 수신한다.

입력은 AJAX를 사용하여 비동기식으로 서버측 코드(PHP)로 전달되고 이 서버측 코드는 입력을 데이터베이스에 배치한다. 이전에 언급한 바와 같이 게임 클라이언트 프레임워크는 JavaScript와 비동기식 개체 작성을 지원하는 웹 브라우저를 필요로 한다.

플레이어 입력을 처리하는 것 외에 게임 클라이언트는 플레이어 생성과 제거를 담당한다. 플레이어 생성 시 해당 플레이어의 입력을 추적하기 위해 게임에서 사용되는 고유의 세션기반 플레이어 ID를 생성한다. 데이터베이스에 있는 각 AJAX 콜과 엔트리에는 플레이어의 고유 ID가 포함되어 있다.

또한 게임 클라이언트는 게임 재시작 통지를 접수한다. 이러한 통지가 수신되면 게임별 코드는 적절한 조치를 취하게 된다(예: 새로운 게임에 참여하거나 사용자에게 초기 화면을 제

공하여 새로운 플레이어를 생성함).

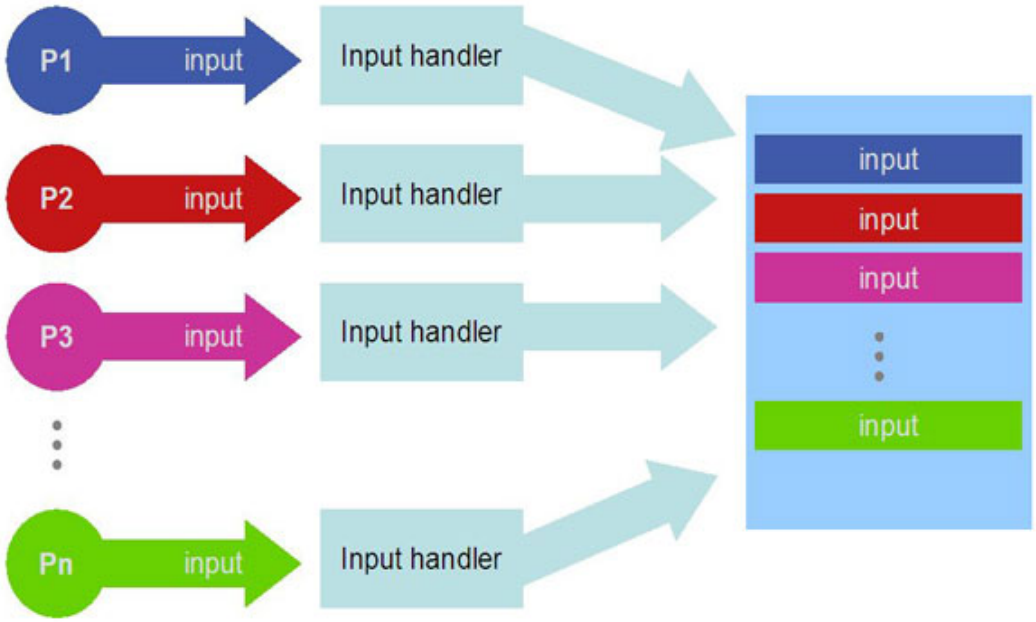

그림 3 데이터베이스에 대한 플레이어 입력 흐름

게임 서버

게임 서버는 게임 클라이언트 프레임워크를 보완해 주는 역할을 한다. 게임 서버는 데이터베이스에서 플레이어 입력을 검색하는 서버 코드 및 게임 UI로 이루어져 있다. 게임 서버 코드는 게임 클라이언트 코드처럼 작동하지만 게임 서버가 극복해야 할 주요 문제는 사용자 입력을 처리하고 게임 공유 스크린을 최소한으로 지연시켜 업데이트하는 것이다.

웹 애플리케이션의 경우 서버에서 업데이트를 검색하는 일반적인 방법은 설정된 간격으로 서버를 폴링하는 것이다. 우리 팀에서는 플레이어의 입력이 플레이어측에서 공유 게임 뷰까지 가능한 신속하게 이동되는 것으로 보이는 환경을 생성해야 했다. 서버를 콜할 때마다 비동기식 개체가 생성되므로 비동기식 콜을 사용하여 서버를 폴링하는 것은 효과적인 방법이 아니다.

그 대신 기존 개체를 잘 활용하기 위해 비동기식 콜에 소규모 트위스트를 적용한다. 서버에 콜을 했을 때 비동기식 개체는 5개 상태를 통과하여 마지막 상태에서 결과가 처리되고 개체는 제거된다. 또한 서버 스크립트와 비동기식 개체 간의 협력을 생성하여 결과를 계속 검색한다.

서버 측에서 스크립트를 생성했는데 이 스크립트는 플레이어의 입력을 루프에서 검색하고 각 반복에서 출력 버퍼를 플러시한다. 출력 버퍼를 플러시하면 마지막 상태에 도달할 때까지 기다릴 필요 없이 현재 결과를 비동기식 개체에 적용할 수 있다. 일단 서버측 루프가 완료되면 비동기식 개체는 이 개체를 제거하거나 재생성하는 것에 지나지 않는 마지막 완료 상태로 변경된다. 이러한 설정과 함께 단일 비동기식 개체로 플레이어의 입력을 검색하기 위해 서버에 대한 다중 콜을 시뮬레이션할 수 있다.

게임 서버의 또 다른 주요 기능은 게임 인스턴스를 유지하는 것이다. 게임 서버는 게임의 생성과 제거를 처리한다. 플레이어가 사용하는 게임 인스턴스 ID는 게임 서버에서 생성되고 플레이어는 데이터베이스를 통해 사용할 수 있다. 또한 게임 서버는 데이터베이스에서 플레이어 입력을 검색한 후 처리한다.

예: 탱크 게임 구축

이전 섹션에서 설명한 프레임워크의 상단에 탱크 게임 및 그룹 조각그림 맞추기 등 두 가지 게임 프로토타입을 구축했다.

이 섹션에서는 탱크 게임을 구축한 방법을 설명한다. 이 프레임워크에서 프로토타입을 구축한 방법을 제시하여 프레임워크의 상단에서 게임을 구축할 수 있는 방법을 보여 주려고 한다. 탱크 게임, 조각그림 맞추기 및 프레임워크는 Intel Software Network 사이트에서 소스 코드 형태로 사용할 수 있다.

탱크게임 클라이언트 코드

탱크 게임에 대한 클라이언트측 코드는 플레이어 생성, 플레이어 액션 및 게임 재시작을 처리한다.

플레이어 생성

플레이어가 스크린의 브라우저를 열면 그림 4와 같이 간단한 HTML 형식이 표시된다. 여기를 클릭하여 HTML 소스 코드를 다운로드할 수 있다.

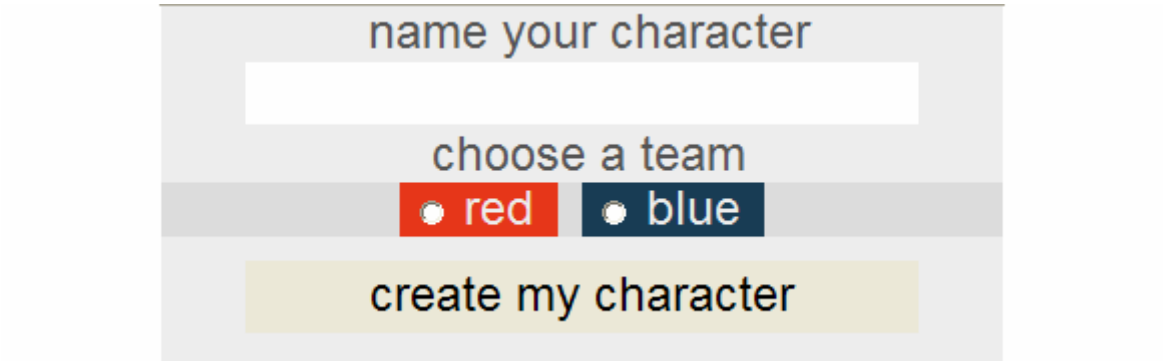

그림 4 플레이어 생성 형식

플레이어 생성 형식이 POSTed이면 PHP 코드는 플레이어 이름과 팀 컬러를 추출한 다음 게임 프레임워크를 사용하여 플레이어에 대한 고유 번호를 식별하고 그림 5의 코드와 같이 새로운 플레이어 인스턴스를 게임 데이터베이스에 추가한다.

```
public static function createPlayer( $name, $team ) {
        // get number for player
        $mates = $GLOBALS['db']->select(array(
            'table' => 'players',
            'fields' => array( 'id', 'teamNumber' ),
            'condition' => array( 'team' => $team )
           ));

           $number = 0;
           if( !empty($mates) ) {
                 $next = $mates[0]['teamnumber'];
                 foreach( $mates as $mate ) {
                           if( $next < $mate['teamnumber'] ) {
                                 $next = $mate['teamnumber']; }
                      }
                      $number = ( $next + 1 ) % self::$NUM_TANKS;
            }

           // create player
           $_SESSION['me'] = new Player( $name, $team, $number );
           // join the game
           $_SESSION['me']->join( $GLOBALS['game'] );
```

그림 5 게임 데이터베이스에 플레이어 추가

현 시점에서 플레이어가 생성되어 게임에 액션(명령)을 입력할 수 있다.

플레이어 액션

플레이어 액션 또는 명령은 플레이어 인스턴스를 성공적으로 생성한 후 클라이언트 브라우저에서 실행 중인 JavaScript를 통해 캡처된다. 그림 6과 같이 이 코드는 우선 입력과 게임 서버(플레이어 명령을 표시한 것이 포함되어 있음)에 대한 POST 메시지를 분석한다(이 코드는 POST 메시지에 대한 jQuery JavaScript 라이브러리를 사용한다).

```
// Player Controls:
// these are mappings to keyboard keys
var Control = {
        Forward: 38, // arrow key up
        Back: 40, // arrow key down
        RotateLeft: 37, // arrow key left
        RotateRight: 39, // arrow key right
        Shoot: 32, // space bar
        Quit: -1
};
// _execute is called when a user presses a key
var _execute = function( _opts ) {
        switch( _opts.keyCode || _opts.which || _opts.other ) {
                case Control.Forward: // move forward
                        _action.type = Command.Move;
                        _action.message = 'forward';
                        break;
                case Control.Back: // move back
                        _action.type = Command.Move;
                        _action.message = 'back';
                        break;
                …
                default:
                        _action.message = '';
         }

 if( _action.message != '' ) {
                // send player command to server
                $.post( _url, _action, function( gameID ) { … }
}
```

그림 6 - JavaScript에서 플레이어 명령 전달.

현 시점에서 클라이언트 코드는 거의 완료된다. 플레이어를 생성하여 플레이어 명령을 게임 서버로 전달한다. 나머지 작업은 게임이 완료되거나 재시작될 때를 감지하는 것이다.

게임 재시작

게임 프레임워크를 통해 게임이 종료되거나 재시작될 때를 간편하게 감지할 수 있다. 플레이어의 명령이 입력될 때마다 게임 서버는 현재 게임 ID로 응답한다. 이 ID의 변경 여부를 확인하고 변경된 경우 createPlayer를 다시 콜한다(그림 5 참고)

탱크 게임 서버 코드

탱크 게임과 조각그림 맞추기 게임의 서버 코드의 경우 공유 스크린에서 Adobe Flash*를 사용하여 이미지와 애니메이션을 렌더링하였다. 본 섹션에서는 탱크를 전진하게 해달라는 플레이어의 요청을 받아들인 사례와 함께 게임 프레임워크와 Flash를 인터페이스로 연결하여 처리하는 방법을 기술한다.

이 프레임워크에 구축된 많은 게임이 Adobe Flash를 사용할 수 있는 반면 게임 프레임워크 코드는 Flash 사용을 허용하지 않음에 유의해야 한다.

게임 프레임워크와 Flash를 인터페이스로 연결

Flash는 게임에 애니메이션 효과를 주는 등 편리한 기능을 제공하지만 게임 프레임워크에서 Flash 인터페이스가 아닌 JavaScript/PHP 인터페이스를 제공하는 문제에 직면하게 된다. 그러므로 주요 도전과제는 JavaScript/PHP에서 Flash로 크리에이션 및 액션 명령을 내리는 방법을 개발하는 것이다. 즉, Flash ActionScript 기능을 내보내기하여 게임 서버의 JavaScript에서 사용할 수 있도록 설정해야 한다. 게임 서버의 JavaScript는 HTTP POST/GET를 통해 PHP 코드와 상호작용한다. 이상의 내용을 구체화하려면 그림 7과 같이 게임 액션 스크립트에서 내보내기한 코드가 다양한 기능을 콜하도록 한다.

```
// set up all external calls
ExternalInterface.addCallback( "addTank", this, addTank );
ExternalInterface.addCallback( "removeTank", this, removeTank );
ExternalInterface.addCallback( "commandTank", this, commandTank );
ExternalInterface.addCallback( "newGame", this, newGame );
ExternalInterface.addCallback( "restartGame", this, restartGame );
```

그림 7 - ActionScript 기능 내보내기

그림 8과 같이 일단 ActionScript 인터페이스를 내보내기하면 Flash 오브젝트의 “CallFunction” 메서드를 콜하면서 인수와 함께 ActionScript 기능을 나타내는 XML 스트링을 패키지로 콜할 수 있다.

```
var _invoke = function( cmdName, args ) {
        var _cmd_str = [
```

```
                '" returntype="javascript">',
                " + _toXML(args) +
                ",
            "].join('');
        return _flashRef.CallFunction( _cmd_str );
}
```

그림 8 - JavaScript에서 ActionScript 콜하기

그림 9(탱크의 전진)와 같이 JavaScript에서 ActionScript를 콜할 수 있는 경우 플레이어 명령을 ActionScript 콜로 매핑하는 문제가 남게 된다.

```
react: function( opts ) {
        var cmd = '';
        var params = '';
        var _message = '';
        switch( parseInt(opts.type) ) {
                …
                case 2: // move
                        cmd = 'commandTank';
                        params = [ opts.source_id, 0,
                                opts.message ].join('_');
                        break;
                …
        }

        if( cmd != '' ) {
                var _ret = _invoke( cmd, params );
        }
},
```

그림 9 - JavaScript 플레이어 액션 명령 파싱

결론 및 향후 작업

멀티플레이어 게임에 대한 새로운 사용 모델을 제시했다. 이 모델은 단일 장소에서 공유 스크린을 보며 모바일 장치로 게임을 컨트롤하는 플레이어들을 활용한 것이다. 웹 개발에서 AJAX와 같은 기술과 아이디어를 차용하여, 사용 모델을 구현하기 위해 게임 프레임워크의 중요한 측면을 설명했다. 이 프레임워크에 구축된 게임을 자세히 살펴 보았다.

마지막으로 해야 할 일은 코드를 공개적으로 사용할 수 있게 하고 게임에 적용했으나 다른 영역으로도 확장할 수 있는 사용 모델을 한번 활용해 보는 것이다. 다음 단계는 귀하의 몫이다. 즉, 코드를 활용하여 자신만의 게임 또는 애플리케이션을 만들어보라. 코드는 Intel의 웹사이트(이메일 주소 필요) 또는 Gamasutra를 통해 다운로드할 수 있다. 자세한 내용은 여

기서 볼 수 있는 비디오를 참고하라.

해당 게임은 사용 모델에 대한 한 가지 사례일 뿐임에 유의하라. 다음과 같이 다양한 향후 작업이 가능하다.

- 기존 게임을 포팅하여 이 프레임워크와 사용 모델을 사용한다.
- 사용 모델을 기반으로 새로운 게임 또는 장르를 개발한다.
- 성능과 응답 시간 향상에 초점을 맞춘다.
- 클라이언트에서 사용할 수 없는 서버의 그래픽 기능에 초점을 맞춘다.
- 더 많은 장소에서 동시에 더 많은 플레이어로 작업해 본다.
- 시청자 투표, 그룹 창의성 및 귀하가 생각할 수 있는 그 밖의 것 등 게임이 아닌 다른 용도에 이 기술을 적용해 본다

이 기술이 계속 활용됨을 듣거나 보게 되면 흥분된다. 이 코드를 사용하고 있다는 소식을 듣게 되기를 바란다. 우리에게 연락하라!