

研修発表

「月着陸船ゲーム+」を作ったことについて

三浦 裕貴

目次

- 実装機能について
- 機能 1
- 機能 2
- 感想と反省

1. 実装機能について

• 1. ユーザーアカウント

- ・ 自動サインアップ・自動ログイン
- ・ アカウント引き継ぎ(**Googleアカウント**)

• 2. スコア関係の機能

- ・ スコアの登録
- ・ プレイログ
- ・ スコアランキング

主に利用したもの

UnityWebRequestを利用

ローカル環境には**XAMPP**を利用

AWS:EC2と**RDS**を利用

サーバーサイドは

PHP—CodeIgniterを利用

機能 1

2. ユーザーアカウントの実装

●仕様段階で少しつまづく

ユーザーアカウントは何が必要か？どう作っていくか？

PCオンラインゲームはどうなってるか

スマホゲームはどうなってるか

Webサービスはどうなってるか



●仕様段階で少しつまづく

ユーザーアカウントは何が必要か？どう作っていくか？

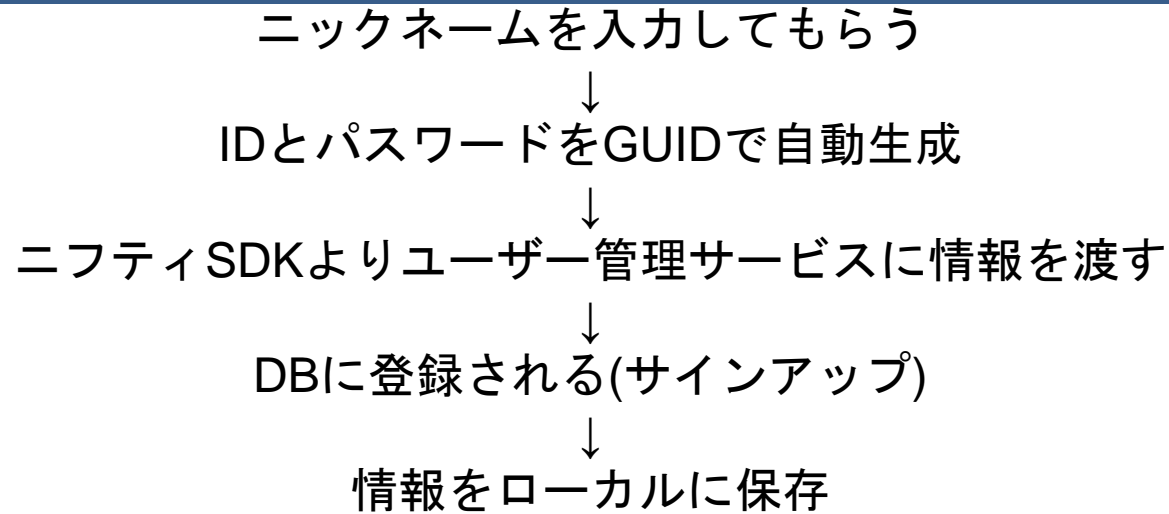
PCオンラインゲームはどうなってるか
スマホゲームはどうなってるか
Webサービスはどうなってるか



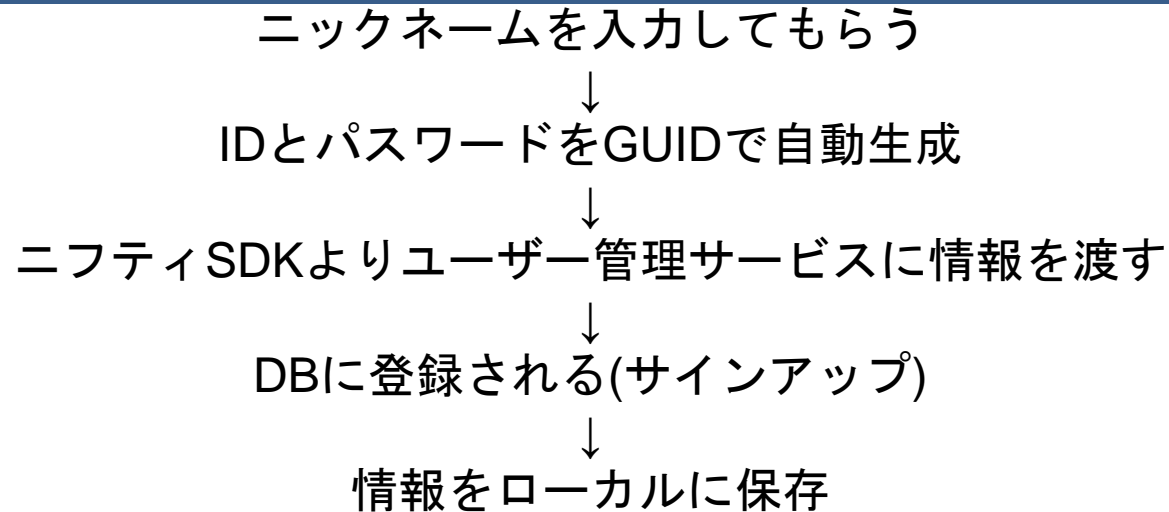
ニフティクラウドを活用した
Unityネットワークゲーム

この本にユーザーアカウントの
自動サインアップ・自動ログイン
について書かれていた

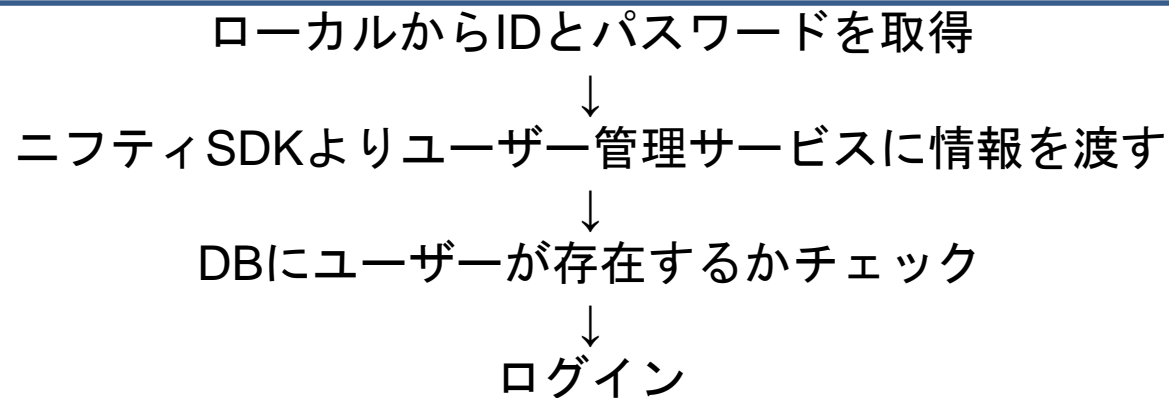
【自動サインアップ】



【自動サインアップ】



【自動ログイン】



【自動サインアップ】

ニックネームを入力してもらう



IDとパスワードをGUIDで自動生成



ニフティSDKよりユーザー管理サービスに情報を渡す



DBに登録される(サインアップ)



情報をローカルに保存

【自動ログイン】

ローカルからIDとパスワードを取得



ニフティSDKよりユーザー管理サービスに情報を渡す



DBにユーザーが存在するかチェック



ログイン

情報をクライアントからサーバーに送信

サーバーが受信 重複チェックやDBに登録 必要な情報をクライアントに送信

クライアントがサーバーから受信

通信部分の処理をUnityWebRequestを利用して作成

アカウント引き継ぎ

Googleアカウントの利用

ゲームからアカウント引き継ぎ用のWebページを開く



Googleアカウントにアクセス



Googleアカウントのプロフィールから

GoogleID(ユニークなID)を取得する。

これを引き継ぎ処理に利用する。

【引き継ぎ設定】

ゲームのユーザーアカウントにGoogleIDを登録する

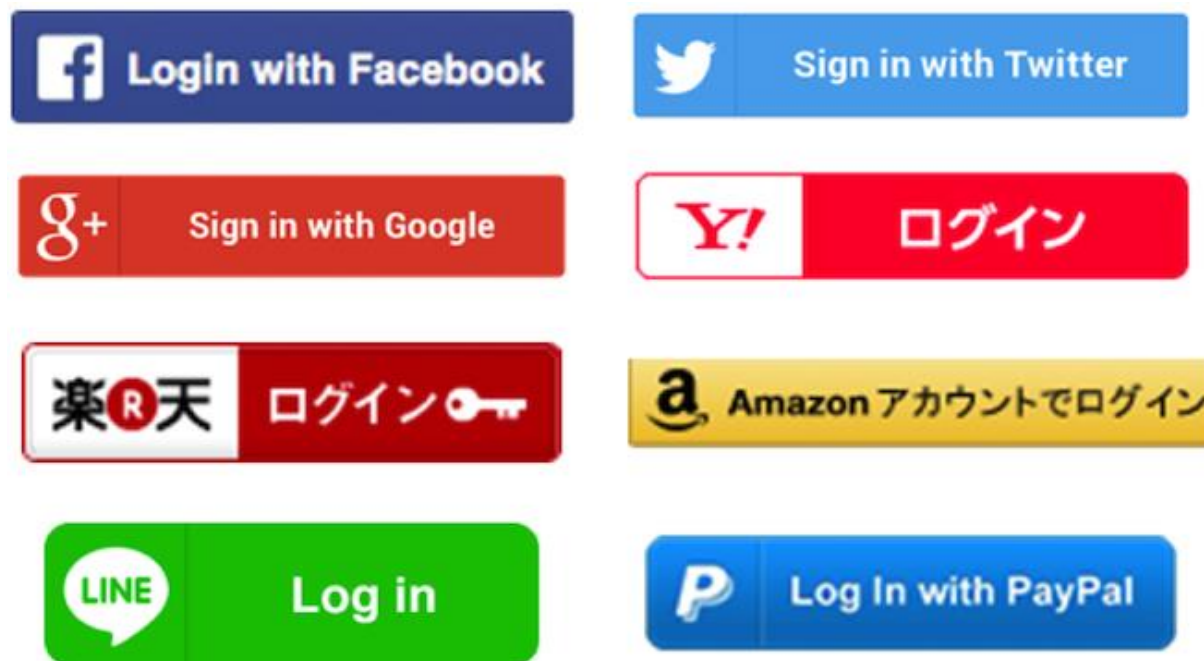
【引き継ぎ処理】

GoogleIDを検索して、

引き継ぐゲームのユーザーアカウントを特定する。

- ソーシャルログイン

SNSアカウントでアプリやWebサービスにログインする
ユーザーのアカウント作成や管理の手間が省ける



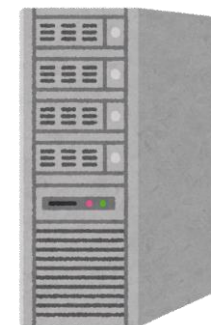
- OAuth(オーオース) 認証
外部サービスがSNSの個人データなどに
アクセスできるように認可する仕組み





ユーザー

認証サーバー



認証クライアント



①プロフィールが見たい

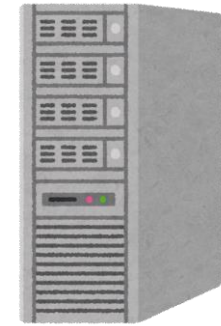
SNS





②プロフィールを見せていい？

認証サーバー



③いいよ

認証クライアント



①プロフィールが見たい

SNS





②プロフィールを見せていい？

認証サーバー



③いいよ

認証クライアント



④アクセス権

①プロフィールが見たい

SNS





②プロフィールを見せていい？

認証サーバー



③いいよ

認証クライアント



④アクセス権

SNS

①プロフィールが見たい

⑤プロフィールが見たい
+アクセス権

SNS

⑥プロフィール

Google側の設定



Google Developers Console

プロジェクトの作成



Google+ APIを有効化

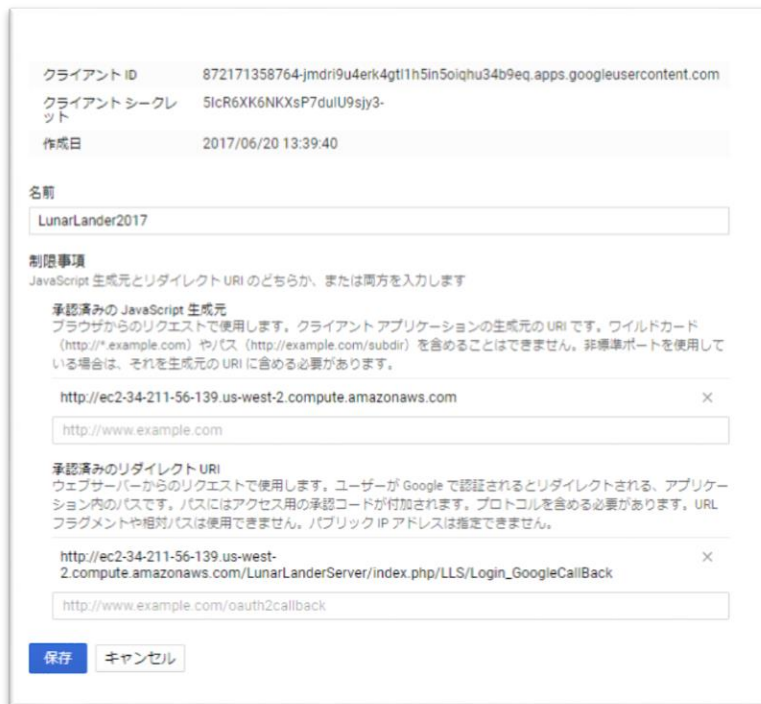


認証情報を作成
リダイレクトURLを設定する
認証クライアントIDと
クライアントシークレット

が生成される



Oauth同意画面の設定
ユーザーに見せるサービス名などの情報



Webページ側の実装



Googleでログインボタンを押す認証ページに飛ぶ

```
/// 認証ページのURLを返す。  
public function getOAuthURL()  
{  
    $baseUrl = 'https://accounts.google.com/o/oauth2/auth?';  
    $scope = array(  
        'https://www.googleapis.com/auth/userinfo.profile', // プロフィール  
        'https://www.googleapis.com/auth/userinfo.email',   // メールアドレス  
    );  
  
    // 認証用URL生成  
    $authURL = $baseUrl .  
        'scope=' . urlencode(implode(' ', $scope)) .  
        '&client_id=' . CLIENT_ID .  
        '&response_type=code' .  
        '&redirect_uri=' . REDIRECT_URL;  
  
    return $authURL;  
}
```

Google側で設定・生成した
クライアントID
承認後開くURL(リダイレクトURL)
を設定する。

```

public function getAccessToken()
{
    $code = $_REQUEST['code'];
    // access_token 取得
    $baseUrl = 'https://accounts.google.com/o/oauth2/token';
    $params = array(
        'code'          => $code,
        'client_id'      => CLIENT_ID,
        'client_secret' => CLIENT_SECRET,
        'redirect_uri'   => REDIRECT_URL,
        'grant_type'     => 'authorization_code'
    );
    $req = new Http_Request2($baseUrl, Http_Request2::METHOD_POST);
    $req->addPostParameter($params);
    $req->setAdapter('curl');
    $res = $req->send();
    $response = json_decode($res->getBody());

    // 失敗した場合はやり直し。
    if(isset($response->error)){
        // トップページに戻る。
        echo 'エラー発生。<a href="'.site_url().'">最初からやりなおす</a>';
        exit;
    }
    // アクセストークンを返す。
    return $response->access_token;
}

public function getUserInfo($access_token)
{
    // ユーザ情報取得
    $userInfo = json_decode(
        file_get_contents('https://www.googleapis.com/oauth2/v1/userinfo?' .
            'access_token=' . $access_token)
    );

    return $userInfo;
}

```

承認後開くページでアクセス権の取得を行う。

クライアントID
クライアントシークレット
リダイレクトURL

アクセス権を利用して
ユーザーのデータを取得する

あとはユーザーデータを利用していただけ

引き継ぎの流れは実際に

引き継ぎの流れは実際に

ID	PASS	NAME	GoogleID
c3a7ff3c	9cc0b0d1	TEST	NULL
47656c42	46dd0c6f	TEST4	NULL
efd734d1	8198f594	TEST2	112259089647109579289
365d81f8	f469329e	USER2	NULL
0b4dedde	68446301	USER3	NULL
f06c9550	10561d2b	USER4	NULL
80b324cb	be4c9ebd	USER5	NULL

機能 2

3. スコア関係の機能実装

スコアの登録

ゲームオーバーになったら、
ユーザーIDとプレイデータ(スコア・クリアステージ数)
を送信する。

サーバーはDB：プレイデータ用のテーブルに
IDとプレイデータ＋日付をどんどん追加していく

```
/// <summary>
/// プレイデータの送信リクエスト
/// </summary>
/// <returns>The play data request.</returns>
/// <param name="playdata">Playdata.</param>
/// <param name="userdata">Userdata.</param>
public static IEnumerator SendPlayDataRequest(PlayData playdata, UserAccountData.UserData userdata)
{
    // リクエストURLを生成
    string url_base = Const.WebRequest.BASE_URL + "SendPlayData/";
    string url_param = "?id="+userdata.id+"&score="+playdata.Score+"&stage="+playdata.ClearStage;
    UnityWebRequest request = UnityWebRequest.Get(url_base+url_param);

    if (WebRequestHeader.HeaderEmpty ()) {
        request.SetRequestHeader (Const.WebRequest.HEADER_NAME_COOKIE, WebRequestHeader.header);
    }

    // リクエスト送信
    yield return request.Send();
}
```

NUM	ID	SCORE	CLEARSTAGE	PLAYDATE
2	efd734d1	430	1	2017-08-25
3	efd734d1	441	2	2017-08-25
4	efd734d1	252	1	2017-08-28
6	c3a7ff3c	969	4	2017-08-28
7	c3a7ff3c	779	3	2017-08-28
8	c3a7ff3c	588	4	2017-08-28
9	c3a7ff3c	126	8	2017-08-28
10	47656c42	670	3	2017-08-28
11	47656c42	316	4	2017-08-28
12	47656c42	961	6	2017-08-28
13	47656c42	870	9	2017-08-28
14	efd734d1	758	10	2017-08-28
15	efd734d1	859	3	2017-08-28
16	efd734d1	707	10	2017-08-28
17	efd734d1	706	3	2017-08-28
18	4dbe345a	106	4	2017-08-28
19	4dbe345a	382	7	2017-08-28
20	4dbe345a	503	4	2017-08-28
21	4dbe345a	310	5	2017-08-28
22	4dbe345a	747	3	2017-08-28
23	365d81f8	120	1	2017-09-04
24	efd734d1	354	2	2017-09-04

プレイログ

ユーザーページを作成し、そこにプレイログを配置
ユーザーページを開く際に、サーバーにリクエスト
プレイデータテーブルからユーザーIDで検索して取得する。

The screenshot shows a 'USER PAGE' for 'USER1'. The page has a dark background with white and yellow text. At the top left is a 'Close' button. At the top center is the title 'USER PAGE' in yellow. At the top right is a 'Ranking' button. The main content area displays 'USER1' in large white text. Below it, 'HighScore' is shown next to the value '162'. Further down, 'MaxClearStage' is shown next to the value '1'. At the bottom left are three buttons: 'Inherit Setting', 'Inheriting', and 'Delete'. On the right side, there is a 'Play Log' section containing a table with two rows of data.

SCORE	CLEAR STAGE	DATE
162	1	2017-09-04
122	1	2017-09-04

スコアランキング

プレイログから切り替えボタンによって
ランキングを表示するようにした。

プレイデータテーブルから
ユーザーIDの重複が無いようにランキングデータを取得



```
// スコアランキングの取得
public function GetScoreRanking()
{
    $userID = $this->session->userdata("id");

    // IDの重複を除いたスコアの降順で取得する。IDから名前を取得する。
    $query = $this->db->query('SELECT users.NAME,playdata.SCORE,playdata.CLEARSTAGE,playdata.ID FROM users,playdata
        WHERE users.ID = playdata.ID AND users.DELETE_FLAG = 0 AND SCORE in
        (SELECT max(SCORE) FROM playdata GROUP BY ID) ORDER BY SCORE desc');
```

プレイヤーからランキングを
取得するSQL文を考えて使っているが、最適化できる

プレイヤーが多くなっていけば
処理にかかる時間も増えると思われるので
SQL文の最適化・高速化について調べて改善していく必要がある。

4. 感想と反省

- 仕様段階での路線変更があったが結果的になんとか形になった
- サーバーとの通信やDBへのアクセスは改良の余地がある。
- サムライのコードを見た後に自分のコードを見て . . .
- 勉強すべき技術・知識の多さ

ご清聴ありがとうございました。