




ARSketch



# ARSketchが目指すもの(デベロッパー)

## <ターゲット>

めんどくさがり屋の、		準備が簡単→zip配布
妄想を実現させたい、		できることが幅広く、具体的
プログラミング未経験者		操作が楽→ボタンポチポチプログラミング

## <ビジョン>

—空が飛べる！敵と戦える！腹筋も割れる！—

小学生からおばあちゃんまで、		操作が楽→ボタンポチポチプログラミング
10分で作れる、		操作が単純→プログラミング & 即公開

キミの妄想。キミの夢を世界に

# ARSketchが目指すもの(ユーザー)

## <ターゲット>

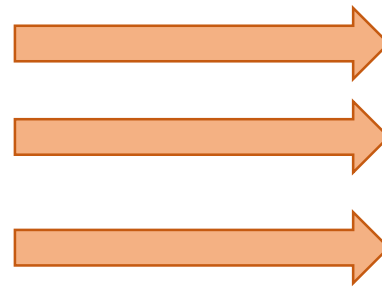
色々な次世代コンテンツを  
今すぐ使いたい人



アプリが多彩・たくさん→あらかじめたくさんのアプリ  
準備が簡単→登録～アプリ起動まで5ステップ以内

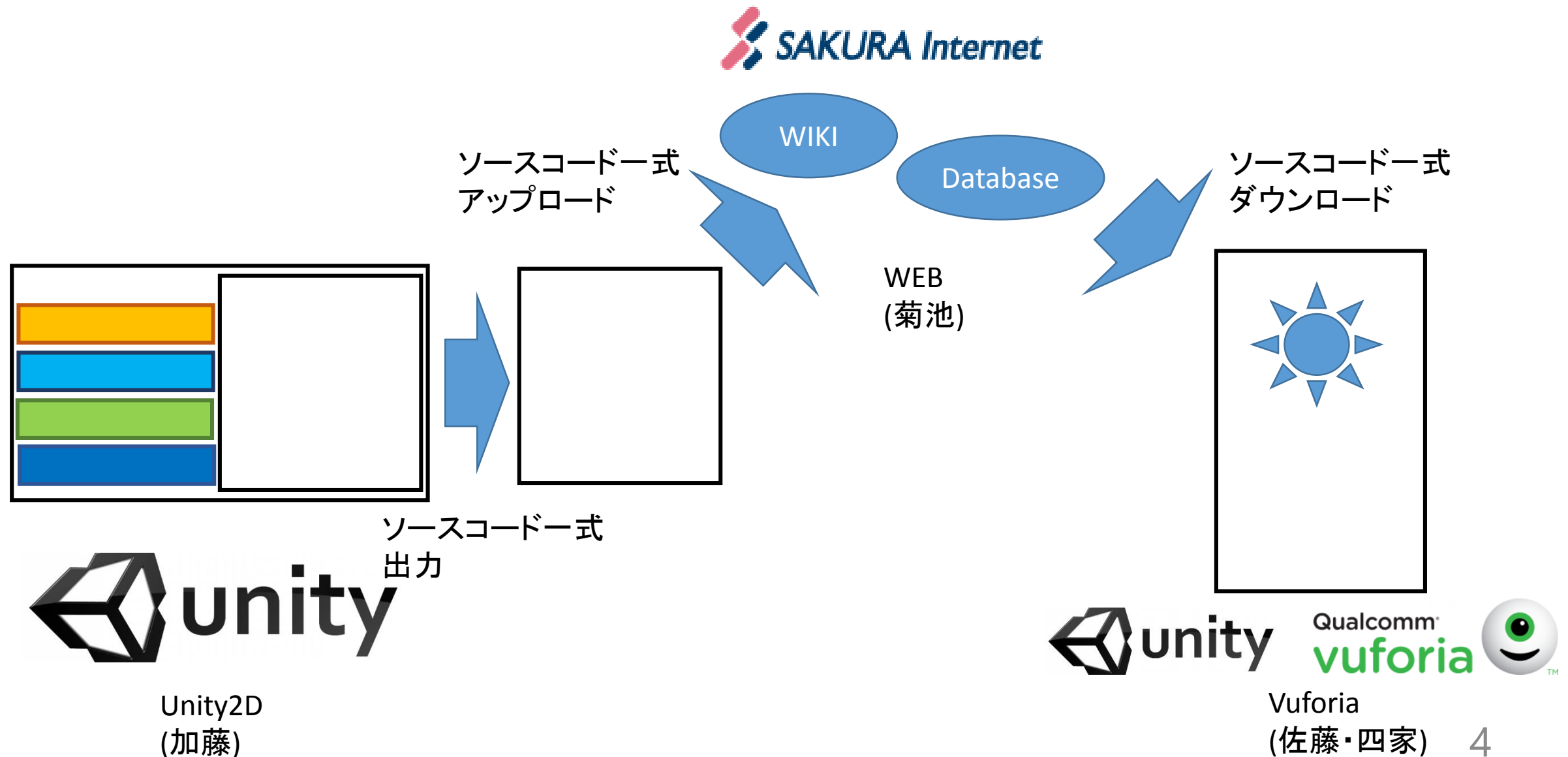
## <ビジョン>

ドラえもんのポケット如く豊富な  
次世代コンテンツを手のひらに。  
夢が並ぶAppストア



テスターにあらかじめ作らせるか？

# 全体構造



# ARSketchE(エディタ)

オブジェクト関係



“MyCube”▽ を “マーカー1”▽ に追加



“MyCube”▽ を “マーカー1”▽ から削除



“MyCube”▽ を “NoName”▽ に変更



モデルを読み込み

Func,Add,“MyCube”,“#1”

Subs,str,Block,“MyCube”

保存

# ARSketchE(エディタ)

## 最低機能

- ポチポチプログラミング  
→文法を覚えなくて良い+エラー処理が楽
- オブジェクトロード
- ジャンル分け
- プロジェクト出力機能
- ウィンドウストレッチ

# ARSketchE(エディタ)

## ○ポチポチプログラミング

“MyCube”▽ を “マーカ-1”▽ に追加



▷ボタンを押すと、対応したソースコードに対応した文を挿入する

```
Func,Add,"MyCube",#1
```

# ARSketchE(エディタ)

## ○エディタ

Func,Add,"MyCube","#1"

一行のインプットフィールドの集まり  
フィールドにフォーカス→文の挿入 の流れ

挿入後、前後に空フィールド(挿入用)を作成する



# ARSketchE(エディタ)

○オブジェクトロード

モデルを読み込み

自分PCに存在する使用可能な3Dモデルを  
プロジェクトフォルダにコピーする

# ARSketchE(エディタ)

○ジャンル分け



挿入ボタンの種類をドロップダウンによって変更できる

挿入ボタンは動的(プログラム上で)に配置しているため、  
何を配置するかを自由に変更できる

# ARSketchE(エディタ)

○プロジェクト出力

保存

ソースコードや3Dモデルを1つのフォルダにして、  
出力する

# ARSketchE(エディタ)

## ○ウィンドウストretch

ウィンドウ内のパーツをウィンドウサイズによって  
柔軟に配置する

→すごいスピードでパーツの場所や大きさを変更している

# ARSketchE(エディタ)

<これまで>

- 1 .Unity2Dの使い方を把握
- 2 .パーツの配置が安定しないので、プログラマ的に配置した
- 3 .挿入ボタンはプレハブ化して生成→中身変更てすれば、  
色々なボタンが作れるはず
- 4 .理想はテキストボックスのマウスカーソルがある行に、  
文を挿入したかったが、カーソル取得方法が  
わからなかったなので1行のテキストボックスを  
無数に並べることにした

# ARSketchE(エディタ)

## <今後の展望>

- パーツ配置を完了させたい
- ジャンル変更によるボタンの再配置
- プロジェクト出力

## 最低機能

- アップロード＋アプリ紹介ページ作成  
→Wiki標準機能で再現可能
- ランキング、マイアプリページ自動作成  
→サーバ上のファイルをプログラミング的に生成することで  
再現可能  
→データベース導入

## <今後の展望>

- ひとまず、アップロード→ページ作成→DB登録の流れを完成させたい

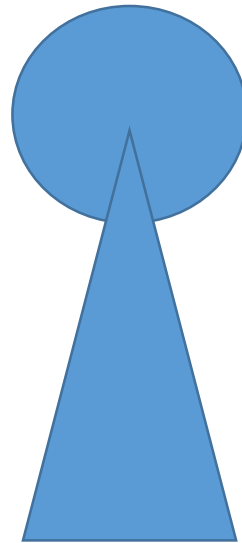


ごめん。まだあまり、検証できてない。

おすすめ

アプリ一覧

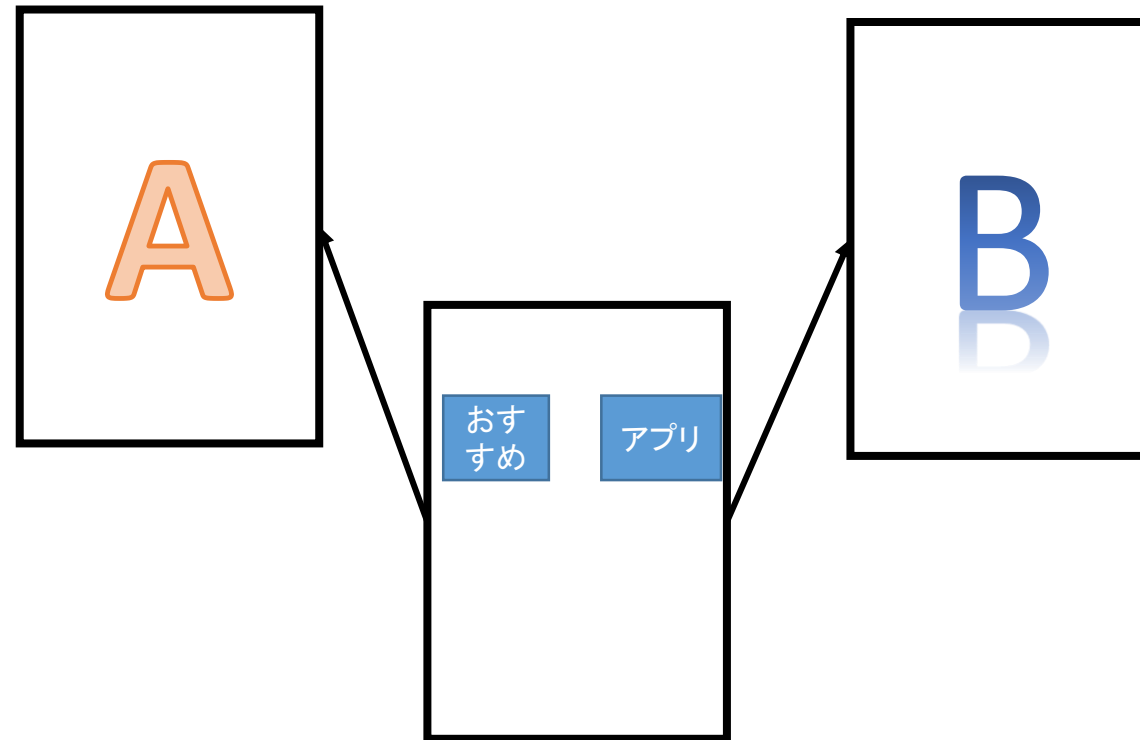
ランキング



## 最低機能

- アプリ起動後、ARアプリを起動できる
- ARアプリのダウンロード

ホーム(四家)→ARアプリ(佐藤)で担当は分けてあるが、  
最終的には1つのネイティブアプリとしてリリース



## <今まで>

1. ImageTargetのデモを試し、ターゲットの子要素にオブジェクトを置けばARができることを理解した
2. オブジェクトを作成し、それをターゲットの子要素にするプログラムを作った
3. イメージマーカーを生成するプログラムを調べた
4. ↑が非常に難しいことがわかった
5. UDTを使い、アプリ上でマーカーを作成することはできた
6. ↑ただし、ユーザーがカメラで写真を撮って、それをマーカーにすることしかできなかつたので、開発段階でマーカーを定義することはできなかつた  
理想は予め画像を用意し、それをマーカーにしたかったのだ

- 7. UDTではダメなので、フレームマーカークを考えた
- 8. フレームマーカークは予め登録された512種類のマーカークである
- 9. 予め登録されているんならイメージでいいじゃないか  
と思うかもしれないが、様々な画像を大量に用意する必要がある上、ユーザーは画像をいちいち印刷する手間がかかる  
フレームならばインクはそんなに必要なく、記号的デザインなのでデザインによる区別がない  
(使うデザインを制限しておけばユーザーも便利)

10. フレームマーカーをプログラミングから生成しようとしたが、何故かできなかった(関数はある)。ので、予め登録しておけばいいと思った。512個だし...
11. コンソールにマーカー1 found とか出てるので、マーカーの検出クラス(関数)を見つけてみた。
12. オブジェクト生成、コンポーネント、プレハブ関連を調べ、まとめた。(別pdf参照)

- 13.別テキストファイルを一行ずつ読み取り、  
それを元にオブジェクトを登録したりするプログラムを作った。
- 14.テキストファイルは便利なのでコマ区切り
- 15.読み取りプログラムは、  
「ここがこうで、ここがこうならこう」という条件分岐の連鎖なので大量のswitch文で木構造にしている



ソースコード例(テキストファイル)

Subs,num,marker,1

Cont,if,eq,@2,"found"

Cont,if,eq,@3,"found"

Func,add,#marker,"mycube"

1列→文の種類(関数Func、制御文Cont、代入Subs)

2列→関数、制御文名および代入の種類(型)

3列目以降→文による

ソースコード例(テキストファイル)

Subs,num,marker,1

[代入,数値,変数名はmarker,値は1]

Cont,if,eq,@2,"found"

[制御文,if文,比較は等値(等しいか),マーカー2,found]

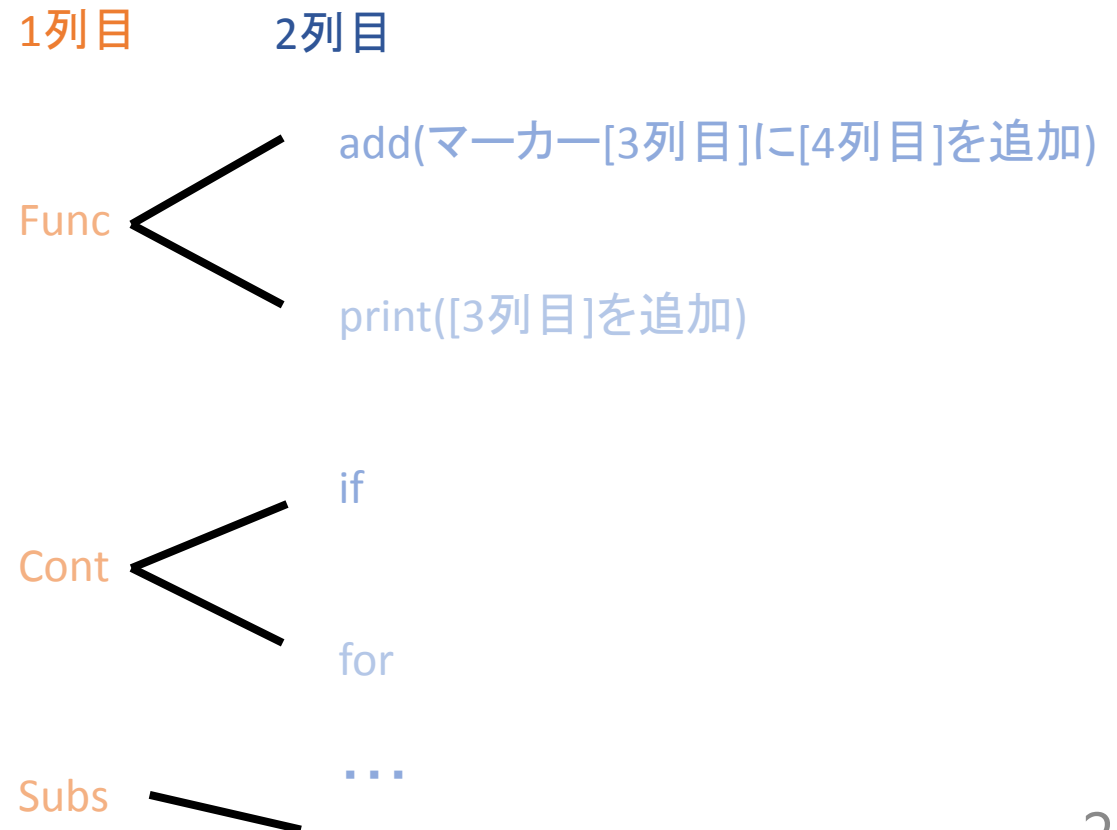
Cont,if,eq,@3,"found"

[制御文,if文,比較は等値(等しいか),マーカー3,found]

Func,add,#marker,"mycube"

[関数,add関数,マーカーmarker,mycube]

```
switch(csvdata[0]){  
  case "Func":  
    switch(csvdata[1]){  
      case "add":  
        . . . . .  
      case "print":  
        . . . . .  
    }  
  case "Cont":  
    . . . . .  
}
```



## if文の仕組み

<ソースコード>

if文1

...

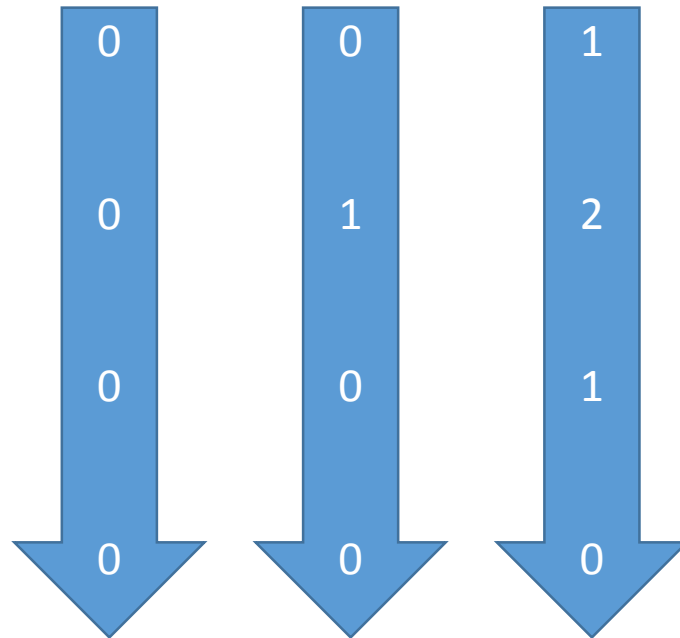
if文2

...

end if2

...

end if1



if文でfalseごとにifカウンタ+1

end ifを踏むごとにifカウンタ-1

カウンタが0でない間は途中の文は無視する

(厳密にはもう少し複雑)

左の文の流れを目で追ってみよう

一番左: true→true

中 央: true→false

一番右: false→false

## ＜今後の展望＞

- ループや演算子、関数を豊富にしたい
  - 手の検出を実装したい
  - Extended Tracking(マーカー外検出)を実装したい
  - Cardboardに対応させたい
- 
- ホーム画面の作成
  - アプリダウンロード機能の実装
- assets/＜アプリ＞/[scriptやresources] と区切るべきか