



株式会社 Cygames
折田 武己

ゲーム開発におけるデバッグ作業の自動化

～OpenCVの「眼」で捉え、Pythonの「脳」が思考し、Appiumの「指」で動かす～





株式会社Cygames
CTO室 ソフトウェアエンジニア
折田 武己

さまざまな基幹系システムの設計・開発を経て、
2012年より株式会社Cygamesに教育担当として参画。
エンジニアを対象にした技術研修やインターンシップの
企画・運営に従事。
アルバイト学生の技術指導を兼ねて、業務効率を
改善するためのツールの設計・開発にも取り組んでいる。



はじめに

Nowtime!
Is the

「受け入れテスト」では、それを実施するのと同
等か？あるいはそれ以上に、エビデンスを保全
することが大切です。

しかし、**（第三者にも検証可能な形で）エビデン
スを残すことは、想像以上にコストのかかる大変
な作業です。**

私の所属する部署では、アルバイト学生の技術指
導も兼ねて「受け入れテスト」を自動化するツール
を開発しています。

本セッションでは、その開発事例について（ノウ
ハウも交えながら）紹介します。



人間による動作検証

そもそも、人間はどのようにしてスマホアプリの動作検証を行っているのでしょうか？

おおまかに切り分けると、下記のような役割分担になるのではないのでしょうか？

眼

- ・ スマホ画面を眺める
- ・ 情報を読み取る

脳

- ・ 読み取った情報を解析する
- ・ 次のアクションを決定する

指

- ・ スマホ画面をタップする
- ・ 情報を入力する



コンピュータによる動作検証

コンピュータに同じことをさせる場合、どのように対応させれば良いでしょう？

私たちは、下記のようにマッピングすることで、問題を解決できると考えました。

眼

• OpenCV

脳

• Python

指

• Appium



技術基盤

技術基盤の各要素について、簡単に紹介します。

- ❖ OpenCV
- ❖ Python
- ❖ Appium



技術基盤【OpenCV】

OpenCVは、オープンソースのコンピュータビジョン向けライブラリです。C/C++のほか、Java、Pythonを公式にサポートしています。

- 画像処理（変換、分割、検出）
- 構造解析
- モーション解析
- パターン認識
- カメラキャリブレーション
- 機械学習
- ユーザーインターフェイス

【URL】 <http://opencv.org>





技術基盤【Python】

スクリプトの開発には、Pythonを利用しています。プログラミング言語としてPythonを選んだ理由は下記の通りです。

- OpenCVとの親和性の高さ
- C/C++のモジュールが利用可能であること
- Numpyをはじめとする優れたライブラリ群

【URL】 <https://www.python.org>





技術基盤【Appium】

Appiumは、スマホアプリを自動操作するためのミドルウェアです。

AndroidとiOSの両方をサポートしており、下記のプログラミング言語でスクリプトを記述することができます。

- Java
- JavaScript
- Perl
- PHP
- Python
- Ruby

【URL】 <http://appium.io>





システム構成

ターゲットとなるスマホOSの種類によって、システムの構成も変わります。Androidの場合、実機を使う以外に事実上の選択肢がありません。

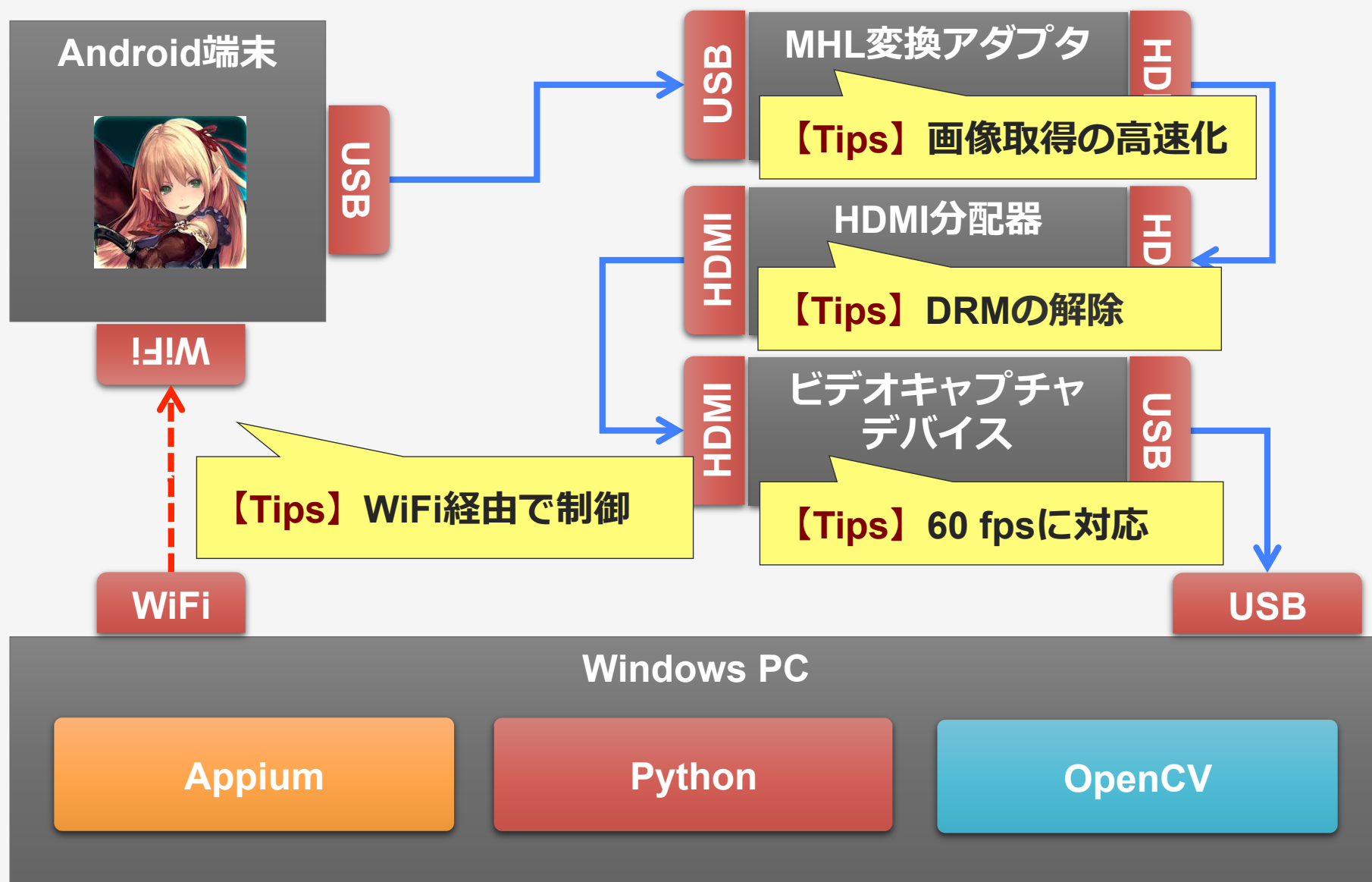
❖ Androidの場合

❖ iOSの場合

「受け入れテスト」の本来の目的に照らし合わせれば、実機を動作検証に使うのは好ましいことです。しかし、**バッテリーの充電速度より消費速度の方が早いため、長時間の連続稼働が難しい**という欠点があります。

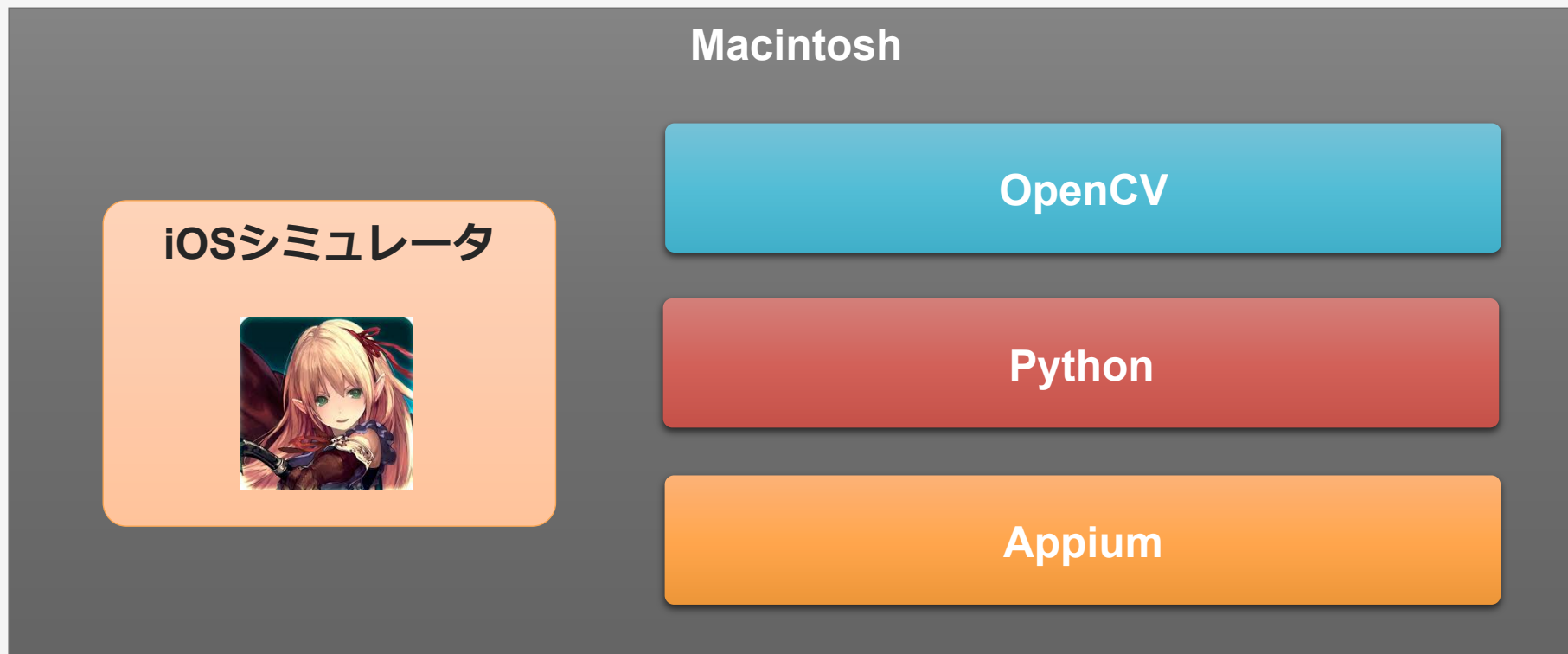


システム構成【Androidの場合】





システム構成【iOSの場合】



iOSシミュレータは実機と遜色のない速度で動作するので、実機を使わずに済ませることも可能です。

Lightning接続のAVアダプタを利用すれば、Androidの場合と同じように実機からHDMI信号を取り出せます。



テンプレートマッチング

OpenCVには様々な機能が用意されています。
テンプレートマッチングは「受け入れテスト」の
自動化に必要不可欠です。

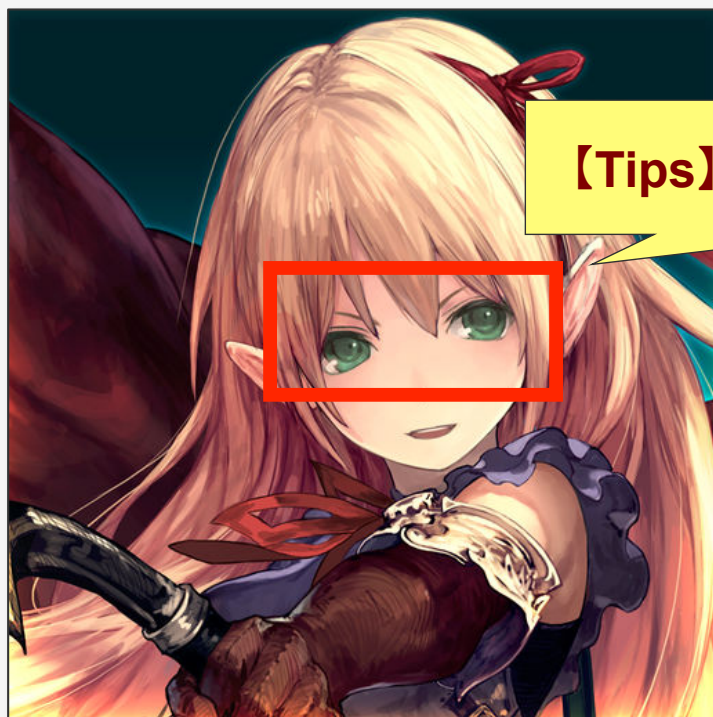
❖ 概要

❖ 特徴



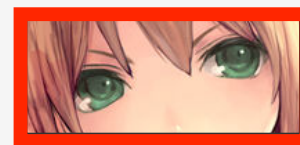
テンプレートマッチング【概要】

テンプレートマッチングを使えば、「ターゲット画像」の中に、指定した「テンプレート画像」が含まれているか？調べることができます。



ターゲット画像

【Tips】 マッチした座標を返却



テンプレート画像



テンプレートマッチング【特徴】

テンプレートマッチングには、下記のような特徴があります。

- ビットマップの完全一致ではない
- 多少のサイズ変動にも対応できる
- 多少の傾きにも対応できる
- マッチングの閾値を指定できる

テンプレート画像を見つけた場合、ターゲット画像中の（複数個マッチした場合、そのすべての）座標を返却します。

画面遷移のトリガーを検出するのに最適です。



基本パターン

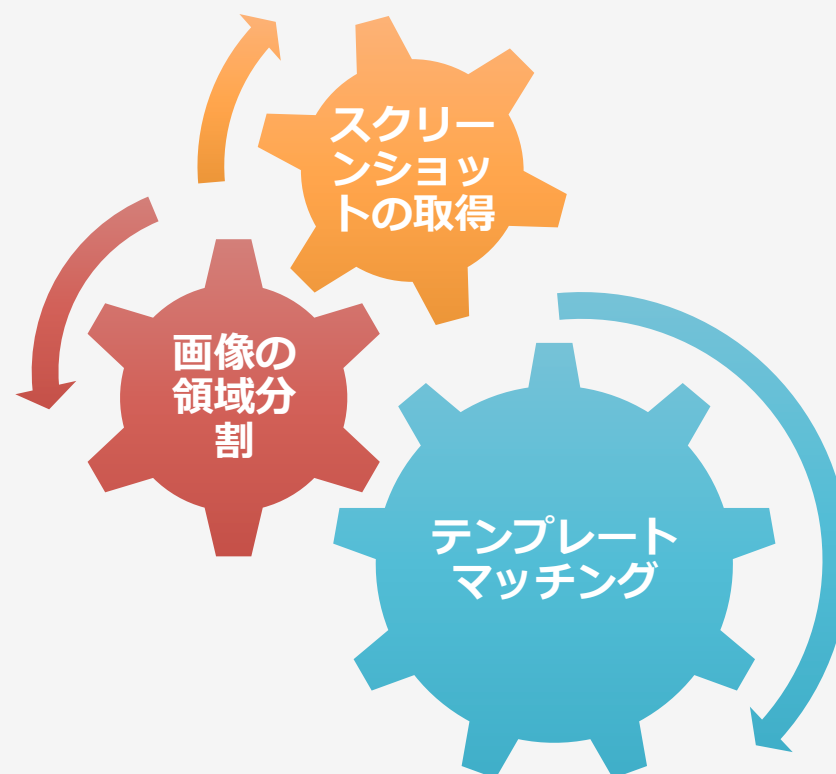
「受け入れテスト」の自動化は、下記の基本パターンを組み合わせることで実現可能です。

- ❖ 状況分析
- ❖ 行動決定
- ❖ 画面遷移
- ❖ 時間調整
- ❖ 情報取得



基本パターン【状況分析】

ターゲットとなるスマホアプリが現在どのような状態にあるのか？スクリーンショットを領域分割し、個々の領域を画像解析することで判断します。





基本パターン【行動決定】

状況分析を行ったのち、どのようなアクションを取るべきなのか？ 各種パラメータや内部状態を勘案しながら、スクリプトが決定します。





基本パターン【画面遷移】

ボタンやリンクをタップすることによって画面遷移を行います。ただし、それらのトリガーとなるオブジェクトの座標を決め打ちするのは、（メンテナンス性を考慮すると）好ましくありません。





基本パターン【時間調整】

次にとるべきアクションが決定しても、それを即座に実行に移せない状況が発生し得ます。

1. データのアップデート中
2. サーバーとの通信中
3. お知らせ等のカットイン
4. エラーの発生
5. その他（想定外）の事象

上記1と2の場合、タイムアウトを考慮した上で一定時間スリープします。



基本パターン【情報取得】

画面に表示された「画像」や「文字」から情報を取得します。描画された「文字」は、ビットマップ化された「画像」に過ぎないので、テキスト化のためのプロセスが別途必要となります。

画面の領域分割

画像のトリミング

データの符号化（OCR / CV）

データの永続化



ケーススタディ

Shadowverseのカードパックの実装事例から、技術的なポイントを解説します。

- ❖ アプリ起動
- ❖ タイトル画面
- ❖ ローディング画面
- ❖ ホーム画面
- ❖ ショップ画面
- ❖ カードパック購入画面
- ❖ カード一覧
- ❖ カード詳細





ケーススタディ【アプリ起動】

Appiumを使ってスマホアプリを起動します。

その際に指定するのが、アプリを一意に特定するための「パッケージ名」です。私たちのシステムでは、必要な情報をJSON形式の定義ファイルに記述します。

Shadowverseの場合、下記のように指定します。

```
"appPackage": "jp.co.cygames.Shadowverse"
```



ケーススタディ【タイトル画面】

タイトル画面が表示されたら、画面の任意の場所をタップしてホーム画面に遷移します。





ケーススタディ【アカウント連携設定】

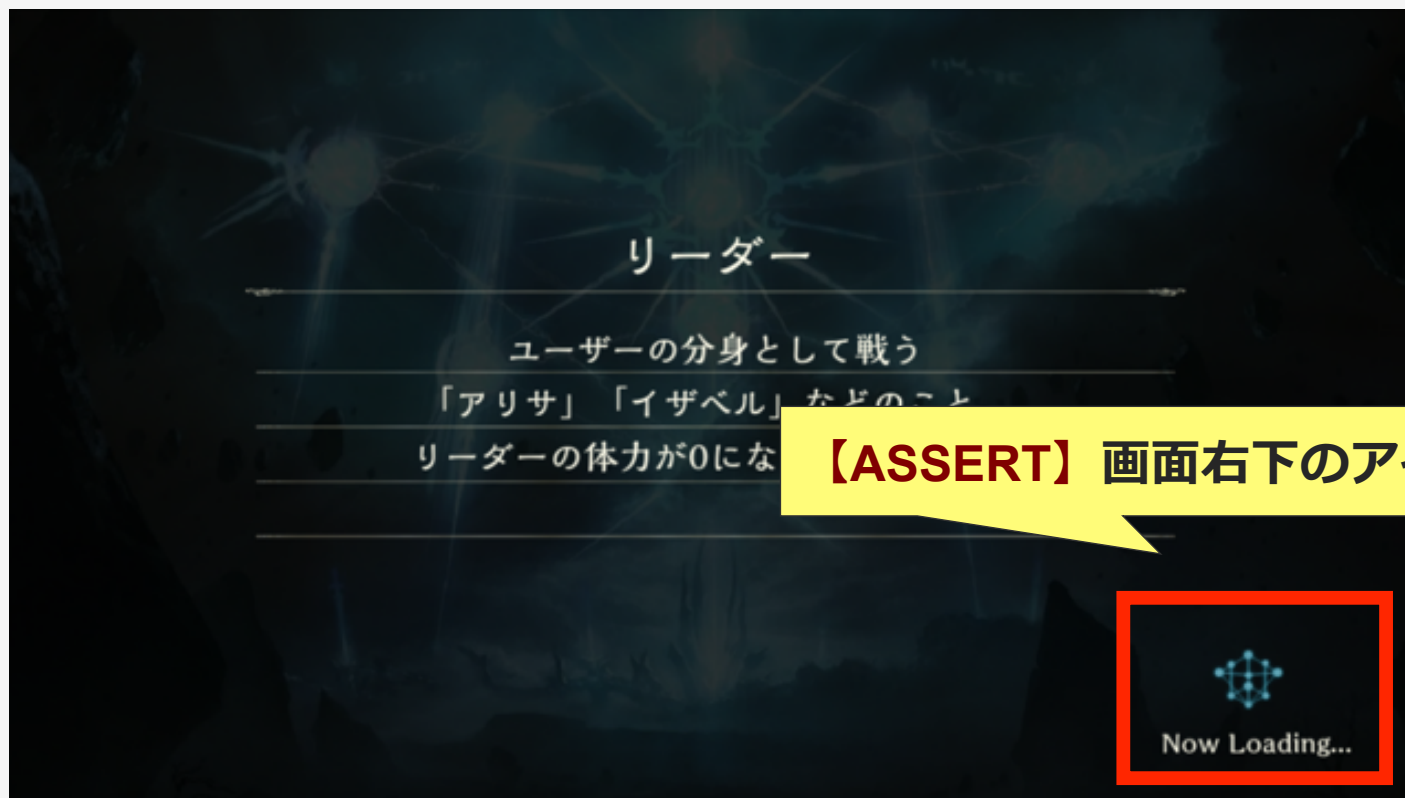
アカウント連携設定に関するダイアログが表示されたら、画面下部の「後で」ボタンをタップして設定を保留します。





ケーススタディ 【ローディング画面】

サーバーとの通信している最中は、ローディング画面が表示されます。タイムアウトを設定した上で、通信が完了するまで待機します。



【ASSERT】画面右下のアイコン



ケーススタディ 【お知らせダイアログ】

お知らせダイアログが表示された場合には、「閉じる」ボタンをタップします。





ケーススタディ 【ホーム画面】

ホーム画面が表示されたら、画面下部のボタンをタップして、ショップ画面に遷移します。





ケーススタディ 【ショップ画面】

ショップ画面が表示されたら、カードバック購入のアイコンをタップします。





ケーススタディ 【カードパック購入画面】

カードパック購入画面が表示されたら、チケットでの「購入する」ボタンをタップします。





ケーススタディ 【数量選択ダイアログ】

数量選択ダイアログが表示されたら、画面下部のボタンをタップします。





ケーススタディ 【購入確認ダイアログ】

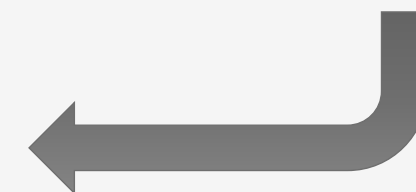
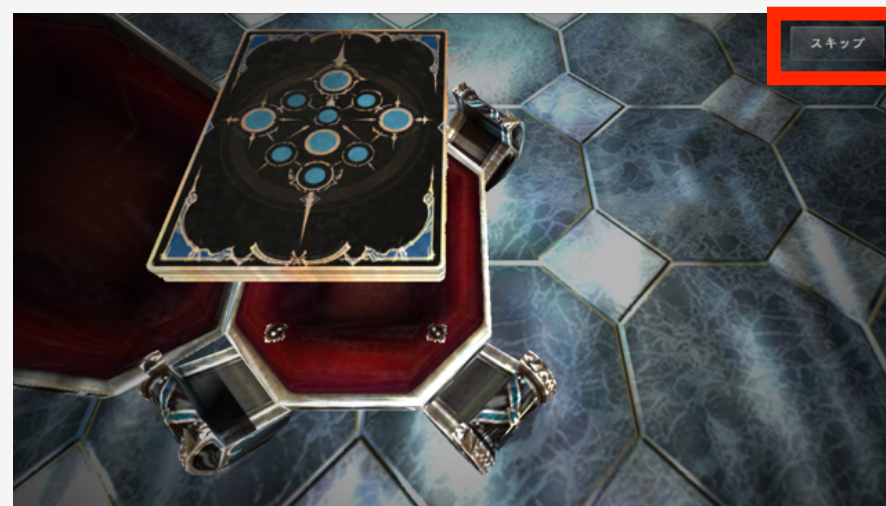
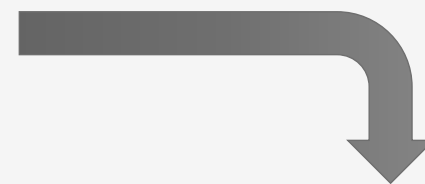
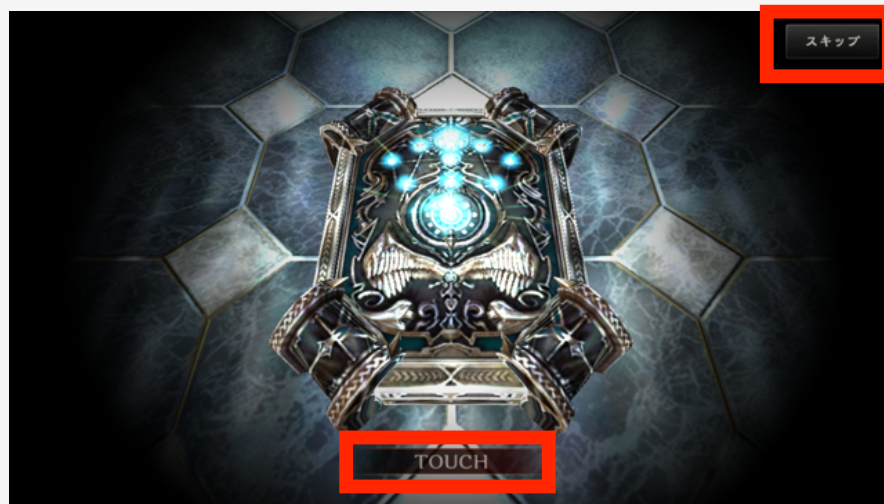
購入確認ダイアログが表示されたら、画面下部の「購入する」ボタンをタップします。





ケーススタディ【アニメーション】

カードパックのアニメーションが表示されます。





ケーススタディ 【カード一覧】

カード一覧が表示されたら、すべてのカードを順番にタップします。





ケーススタディ 【カード詳細：Aパターン】

カード詳細が表示されたら、エビデンスの永続化に必要な情報をテキストデータに変換します。

【ASSERT】画面上部の文字列





ケーススタディ 【カード詳細：Bパターン】

カード詳細が表示されたら、エビデンスの永続化に必要な情報をテキストデータに変換します。

【ASSERT】画面上部の文字列



デモ動画





最適化

テンプレートマッチングの最適化に関するポイントを整理すると、次のようになります。

- ❖ 探索範囲の局所化
- ❖ 画像サイズの調整
- ❖ テンプレート画像の加工
- ❖ テンプレート画像の階層管理



最適化【探索範囲の局所化】

ターゲット画像の全域で照合作業を行うのは計算コストの無駄遣いです。画像をクリッピングすることによって、探索範囲を絞り込みましょう。





最適化【画像サイズの調整】

ターゲット画像やテンプレート画像のサイズが小さければ、その分だけ照合作業を高速に終わることができます。

画像サイズが1/2になれば、占有する面積、つまりメモリ消費量は1/4になる。
照合速度はそれに反比例する。

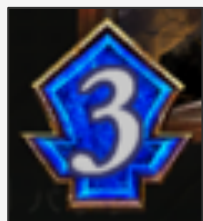


テンプレートマッチングに利用する画像は、エビデンスとして保存する画像と同一サイズである必要はない。



最適化【テンプレート画像の加工】

テンプレートマッチングには「透明色」という概念はありません。テンプレート画像に背景が映り込んでいる場合、照合作業に影響のない範囲でトリミングしましょう。



【NG】 形状が複雑なため、テンプレート画像の中に背景が映り込んでいる。



【NG】 背景部分を除去しても、透明色として扱われない。マスク処理にも対応しない。

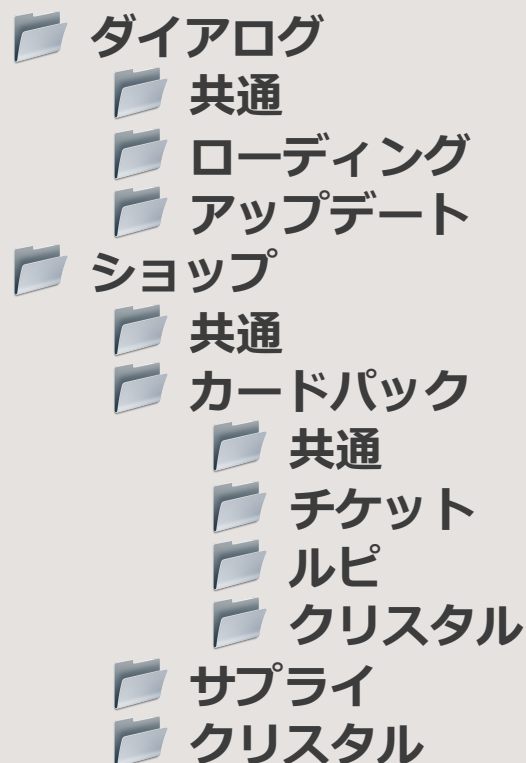


【OK】 背景に影響されないように、テンプレート画像をトリミングする。



最適化【テンプレート画像の階層管理】

照合作業を最少のコストで済ませるためには、テンプレート画像を適切な粒度でグルーピングする必要があります。



テンプレート画像をフォルダごとに分割する。フォルダ同士や同一フォルダ内のテンプレート画像については、出現頻度の高い順に照合作業を行う。



サブシステム

「受け入れテスト」の一部を自動化するにあたり、
下記のようなサブシステムを整備しました。

- ❖ OCRサービス
- ❖ ドライブレーダー
- ❖ エビデンスビューワー



サブシステム【OCRサービス】

市販の日本語OCRライブラリを利用して、ビットマップ画像中に含まれるテキスト情報を抽出するため、下記のような（社内向け）Webサービスを立ち上げました。

- WebSocketによるデータ通信
- 任意の矩形でトリミング
- 任意の矩形でテキスト化領域を指定可能
- 複数のテキスト化領域を指定可能
- 透過背景のノイズを自動除去

個々のPCにOCR機能を組み込む必要がなく、セットアップや運用が容易になります。



サブシステム【ドライブレコーダー】

下記の理由により、スクリプトがタイムアウトもしくは異常終了した場合、直近のn秒間の様子を動画ファイルとして自動保存してくれます。

- ・スクリプトの不備
- ・実機のハードウェア障害
- ・実機のソフトウェア障害
- ・その他（想定外）の事象

スクリプト開発の初期段階では、実装要件の考慮漏れも多く、記録された動画ファイルがエラー原因の解析に役立ちます。



サブシステム【エビデンスビューワー】

スクリプトを実行すると、既定の場所に下記のようなエビデンスが自動保存されます。

- スクリーンショット
- 実行結果のサマリー情報
- 実行結果の詳細情報
- ログファイル
- 例外発生時のスタックトレース
- 例外発生時の動画ファイル

Webブラウザーを使って、社内のどこからでも自動集計されたエビデンスにアクセスできます。また、スクリプトを実行することも可能です。



まとめ

コンピュータビジョンを応用した「受け入れテスト」の自動化は、**すでに実用レベルの域に到達している “開発現場で使える” 技術です。**

細心の注意を払ってスクリプトを実装すれば、熟練したスタッフと遜色のない品質を担保できます。
ただし、人間ほどの柔軟性は備えていないので、想定外の事象にはとても弱いのです。

すべての領域を一括移行するのは、コストやリスクの面からみても現実的ではありません。**反復の頻度の高い処理から順に置き換えるのが、もっとも有効なアプローチだと思います。**



質疑応答

もし疑問や質問があれば、
可能な範囲内でお答えします。