目次

図目次

第 1 章 序論

1.1 本章の構成

第 1 章では，本論文の序論を述べる．研究背景，研究目的，研究方法，プロジェクトマネジメントの関連，本論文の構成について記述する．

1.2 研究背景

1.2.1 現代の情報社会

“コンピューターによる情報システムの利用が人々の生活や企業活動に浸透した現代となっており，私たちの生活にとって無くてはならないものであり，暮らしの中の様々な場面で情報システムが活用されている”[1]．そして，私たち人間はその情報を得る際，その多くを目からにより，様々なものを見て情報としている．特に，スマートフォン・タブレット向けのアプリケーションやサービスの提供が急速に広がりを見せ，それに伴い，スマートフォン・タブレットの利用者も急増している．なぜ，ここまで急増したのか．主な理由として 2 つあると考える．ひとつは，タッチパネルで操作できるということ．従来のボタン操作だと画面をスクロールするには上下左右のボタンを押し続けないと出来ないうえに，少しずつしか画面が移動しなかった．しかし，タッチパネルにすることで，画面のスクロールが指一本でフリップするだけでスムーズに画面移動することができる．もうひとつの理由として，動画サイトが見れるということである．従来の携帯では，動画を表示するには，相応の処理能力と高い通信速度が必要なため，しっかりと見ることができなかった．このようなことから，以前にも増して手軽に写真や画像，動画を見るといった生活になってきていると言える．また，自ら撮った写真をカメラアプリケーションなどを用いて，手軽に編集や画像処理ができるようになった．

1.2.2 画像処理

画像処理とは，デジタル化した写真や絵画，カメラなどで撮った映像などの画像情報を見やすくするために，別の加工・変換したり，その画像の形状や色などの特徴を抽出したり，画像が何を表しているのか認識したりする処理をいう[2]．

1.2.3 画像処理における問題点

スマートフォンやタブレットにより，アプリケーションなどを用いて処理することが容易になってきている．しかし，リアルタイムに動画を処理するアプリケーションを調べたところ見当たらなかった．動いているものを消したり，見やすいものに変えたり，特定の情報だけを取り出して得たりしようとする画像処理技術を出来るわけではない．例えば，一般ユーザーが専用ソフトを使おうとしても，まず言葉の意味が分からないことや，開発途中でうまくいかず断念することがあったりと，画像処理をするのは容易ではない．カメラアプリケーションや画像編集アプリケーションのように，動画を処理するアプリケーションはないかと考えた．

1.2.4 iOS アプリケーションの開発

iOS 専用の開発ツールとして，Xcode を用いる．Xcode は，ソースエディタや視覚的なデバッガなど，優れたアプリケーションを作成するための完全なツールセットである．作成したものを確認するためには，iOS Developer Program への登録が必要である．それにより，実機上でどのように動作するのか確認することができる．この Xcode は，iPhone/iPad には，最も融通が効く開発方法であり実行効率もよく，特に Objective-C を用いた開発部分はメモリ管理が必要である．煩雑な部分も多く，ハードルの高い開発手法である[3].

1.3 研究目的

本研究では，リアルタイムに動画を処理するスマートフォンアプリケーションを開発する．開発するアプリケーションは，スマートフォンやタブレットのベンチマークとなることも期待される．

リアルタイムに動画を処理するスマートフォンアプリケーションは，動いているものを消すということが目的である．そして，このようなアプリケーションを開発することで，いつでも簡単に利用でき，編集する時間と手間が省くことができる．また，開発に必要なツールやプログラミングソースについて，知識を得ることができる．

1.4 研究方法

リアルタイムに動画を処理するスマートフォンアプリケーションの開発をするうえで，iOS の実機上で動かせるようにする．iOS アプリケーションの開発に必要な Xcode で作成する．作成したものは，iPhone Developer Program で登録したのちに，実機上で確認する．このとき，使用するプログラミング言語は Objective-C である．

1.5 プロジェクトマネジントとの関連

アプリケーション開発は PMBOK が提唱する，品質マネジメントに最も関連があるものである．PMBOK における品質管理は，品質計画（計画）・品質保証（実行）・品質管理（管理）という 3 つのプロセスから構成されている．アプリケーション開発において，最初に考えなければならないのは品質基準である．品質状態の把握には，障害発生数だけでなく障害の発生原因や重大度も管理することが大事である．

1.6 本論文の構成

第 1 章では序論，第 2 章では，本論文の研究目的を行う前に，OS アプリケーションの開発環境の構築を記述する．アプリ開発について，必要なソフトはなにか，アプリ開発がどのようなものであり，どのように行われているのかなどを記述する．第 3 章では，アプリケーションの開発に取り組む．ここでは，必要なコードを主に記述する．第 4 章では，プロジェクトマネジメントとの関係を記述し，第 5 章で考察，まとめを行う．

参考文献

[1] 情報化社会の進展. http://itjobgate.jisa.or.jp/about/.

[2] 村上伸一. 学生のための画像処理プログラミング演習. 東京電機大学出版, 2012-4-10.

[3] 臼井洋文. 第 1 回. はじめに. – iPhone/iPad アプリケーション開発方法の各特徴 -. http://gihyo.jp/dev/serial/01/ios-sdk/0001.

第 2 章　開発環境構築

2.1 iOS

Apple 社の携帯情報端末に搭載されている OS(オペレーティングシステム)．同社の iPhone，iPad，iPod touch などに内蔵されている．

同社のパソコン向け OS である Mac OS X の基盤部分に携帯端末向けの機能などを追加して再構成したもので，タッチパネルを前提としたユーザインターフェースなどが特徴となっている．メールや Web ブラウザ，テレビ電話，メディアプレーヤーなど基本的なアプリケーションソフトは予め内蔵されている．同社以外の開発したソフトウェアを使いたい場合は，インターネットを通じて同社の「App Store」からダウンロードして導入する必要がある[3].

2.2 Xcode

Apple 社が Mac OS X 向けに開発・配布している、ソフトウェア開発のための統合開発環境(IDE：Integrated Development Environment)である．

ソースコードの編集やプロジェクトを構成するファイルの管理，コンパイル，ビルド，デバッグなどを行なうことができ，Mac OS X や iOS(iPhone/iPad/iPod)で動作するソフトウェアを開発することができる．標準で使用できるプログラミング言語には C 言語や C++，Java，AppleScript，Objective-C などがある[4].

2.2.1 Xcode のインストール

① アプリケーションフォルダの App Store を起動し，Apple ID とパスワードを入力し「サインイン」を選択する．サインインすると App Store から各種ソフトウェアがダウンロード出来るようになり，「カテゴリ」から「Xcode」を選択する．

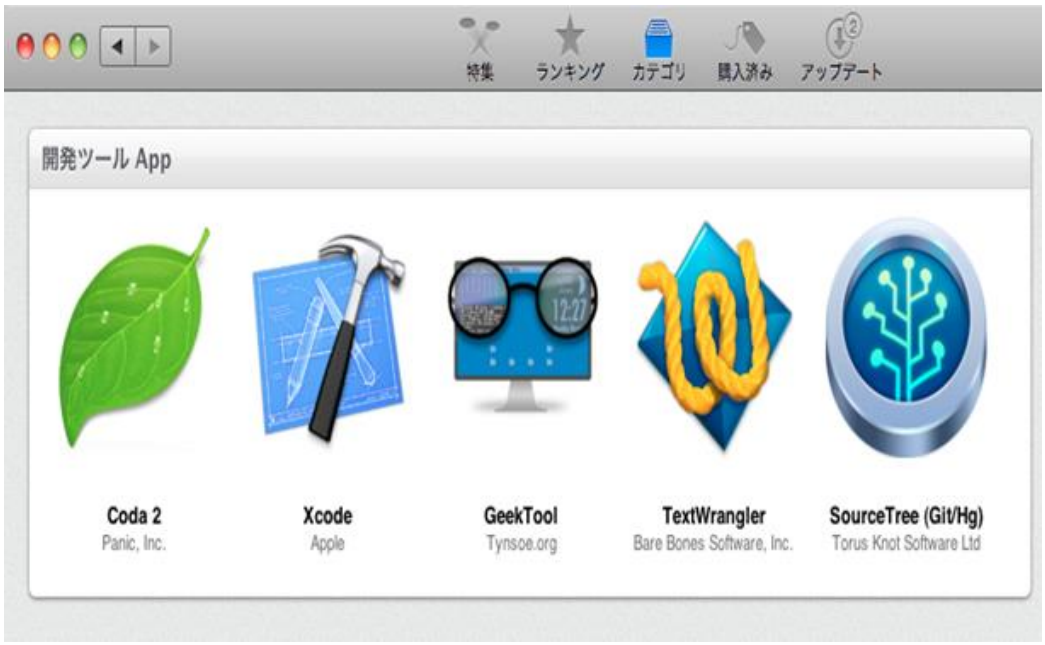

図 2- 1 カテゴリ

② 「無料」ボタンをクリック．インストール処理が始まる．

図 2- 2 Xcode インストール

③ インストールが完了すると「インストール済み」に表示が変わり，サインアウトする．

図 2- 3 Xcode インストール完了

④ アプリケーションフォルダから Xcode をもう一度起動し，Xcode の利用許諾（ライセンス同意契約）の画面が表示されたら，Agree ボタンをクリックする．

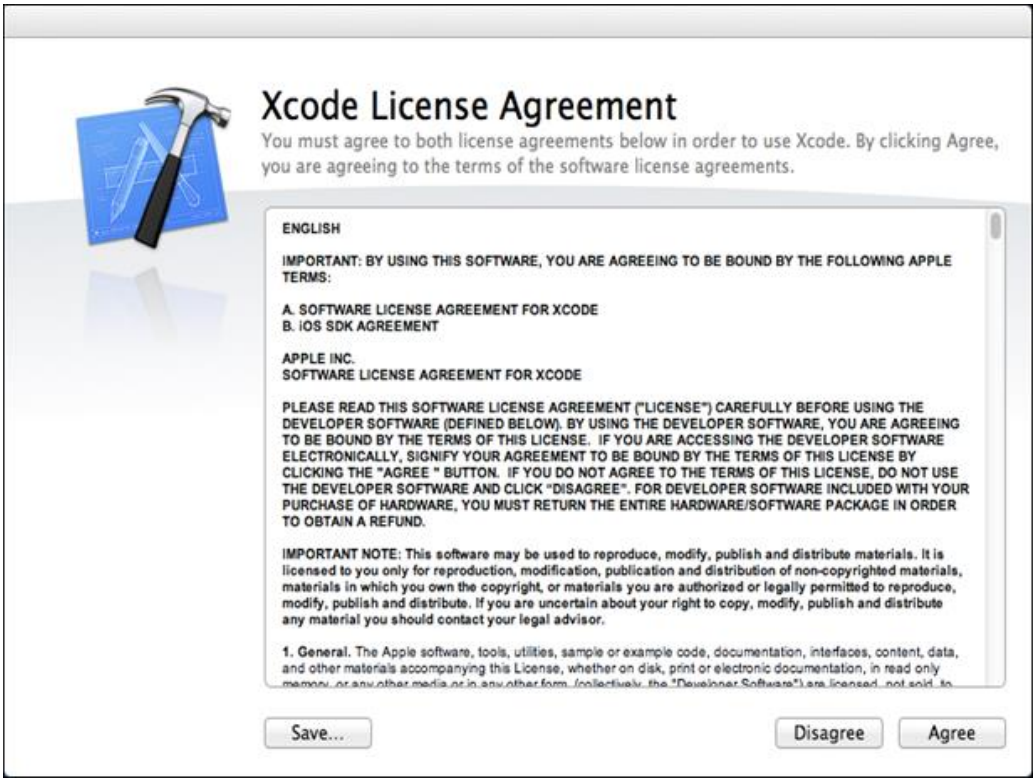

図 2- 4 利用許諾

⑤ Mobile Device Framework のインストールを求められるため，Install をクリックする．

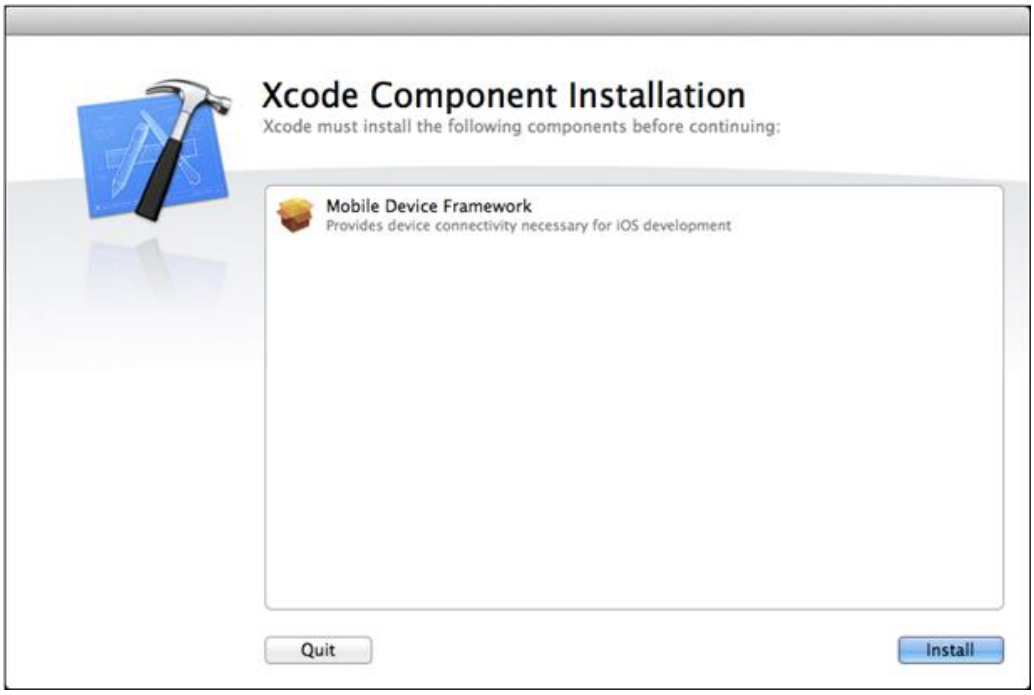

図 2- 5 Mobile Device Framework インストール

⑥ Install をクリックすると，MacBook の管理者権限を要求してくるので，自身の MacBook の管理者パスワード（ソフトウェアアップデート時に使うパスワード）を入力してから，「ソフトウェアをインストール」ボタンをクリックし，ソフトウェアをインストールする．

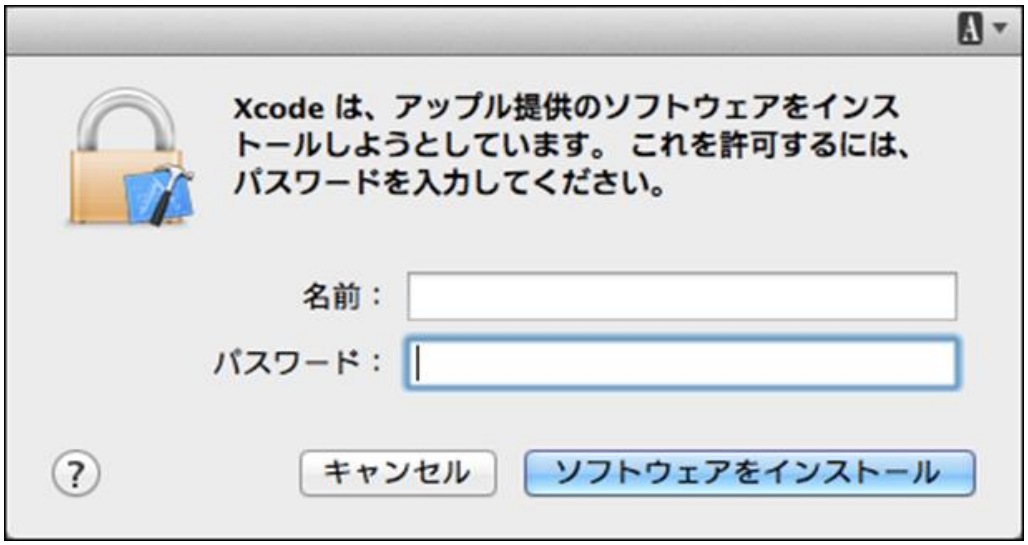

図 2- 6 管理者権限要求

⑦ Mobile Device Framework のインストールが完了したら，Start Using Xcode をクリックすると Xcode が起動する．

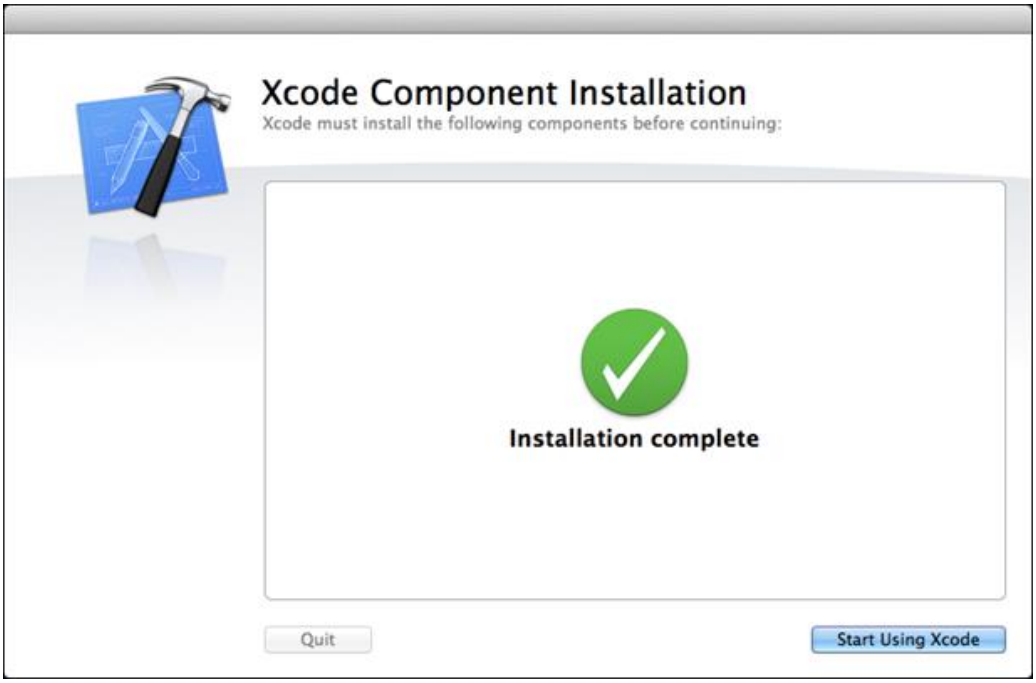

図 2- 7 Mobile Device Framework インストール完了

⑧ Xcode が起動され，「Create a new Xcode project」を選択すると，開発画面へと進む．

図 2- 8 Mobile Device Framework インストール完了

2.3 キーチェーンアクセス

Apple 社が Mac OS 及び Mac OS X 用に開発した, パスワード管理システムのことである. キーチェーンの管理を行うユーティリティは Keychain Access（キーチェーンアクセス）と呼ばれる.

2.4 証明書署名要求(CSR)

デジタル証明書の発行を申請者から認証局に請求するメッセージのことである. 申請者の識別情報や公開鍵などから成り, 秘密鍵で電子署名されて認証局に送信される. CSR を受け取った認証局が証明書の発行を認めると, デジタル証明書を認証局の秘密鍵で署名して申請者に送信する[4].

2.5 キーチェーンアクセスから証明書署名要求(CSR)の作成

「iOS Developer Program」の「Certificates,Identifiers & Profiles」を作る前に, iOS Developer Program の Certificates から証明書ファイルを作成するには, 証明書署名要求(CSR)が必要になります. その為, 最初に Mac のキーチェーンアクセスから証明書署名要求(CSR)を作成する必要があります[7].

① キーチェーンアクセスは，アプリケーションからユーティリティー，キーチェーンアクセスの順で見つかる．今回は，Spotlight に「キーチェーンアクセス」と入力して起動させた．

図 2- 9 キーチェーンアクセス起動

② 証明書署名要求(CSR)を作成するには，キーチェーンアクセスのメニューを開き，以下の順序で選択する．

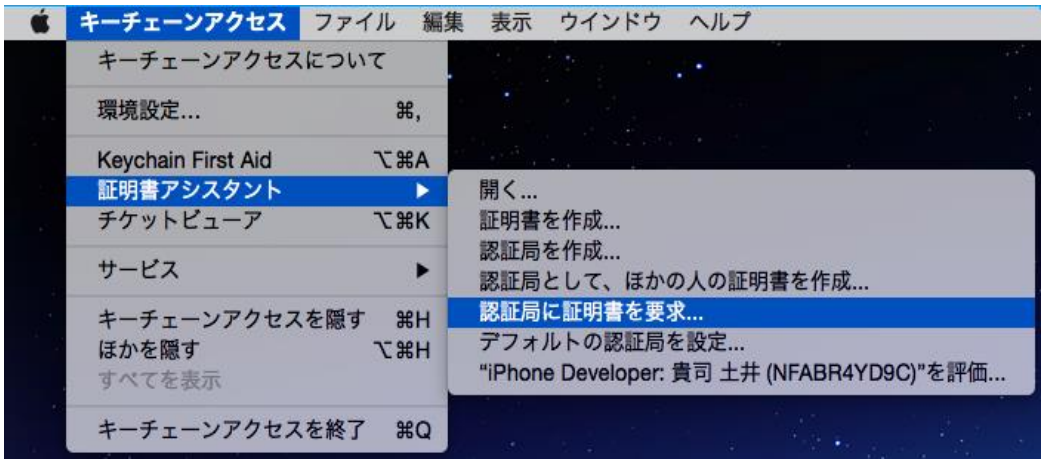

図 2- 10 認証局に証明書を要求

③　証明書情報の入力を行う.

- ユーザーのメールアドレス：iOS Developer Program に登録したメールアドレス
- 通称：iOS Developer Program に登録した名称
- CA のメールアドレス：不要
- ディスクに保存を選択
- 鍵ペア情報を指定：チェック必須

通称は，開発していると証明書が増えてくるため，命名規約を作り，分かり易い通称をつけておくこと.

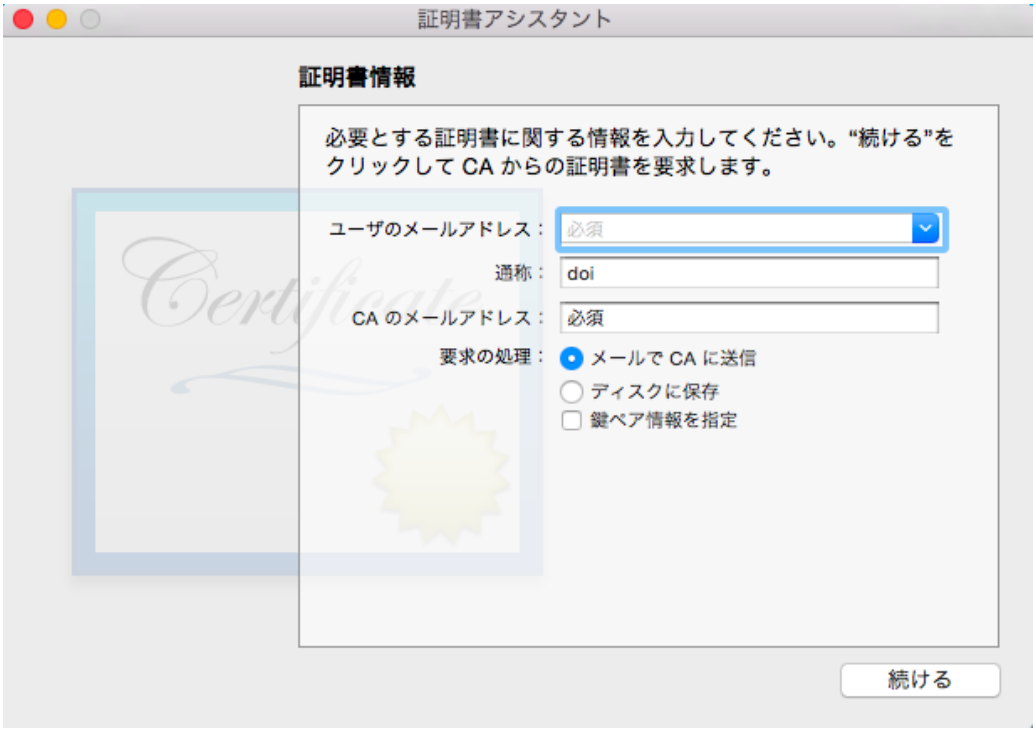

図 2- 11 証明書情報入力

④　鍵ペア情報を選択する.

- 鍵のサイズ：2048 ビットを選択
- アルゴリズム：RSA を選択

鍵のサイズが 2048 ビットでなければ CSR は却下される.

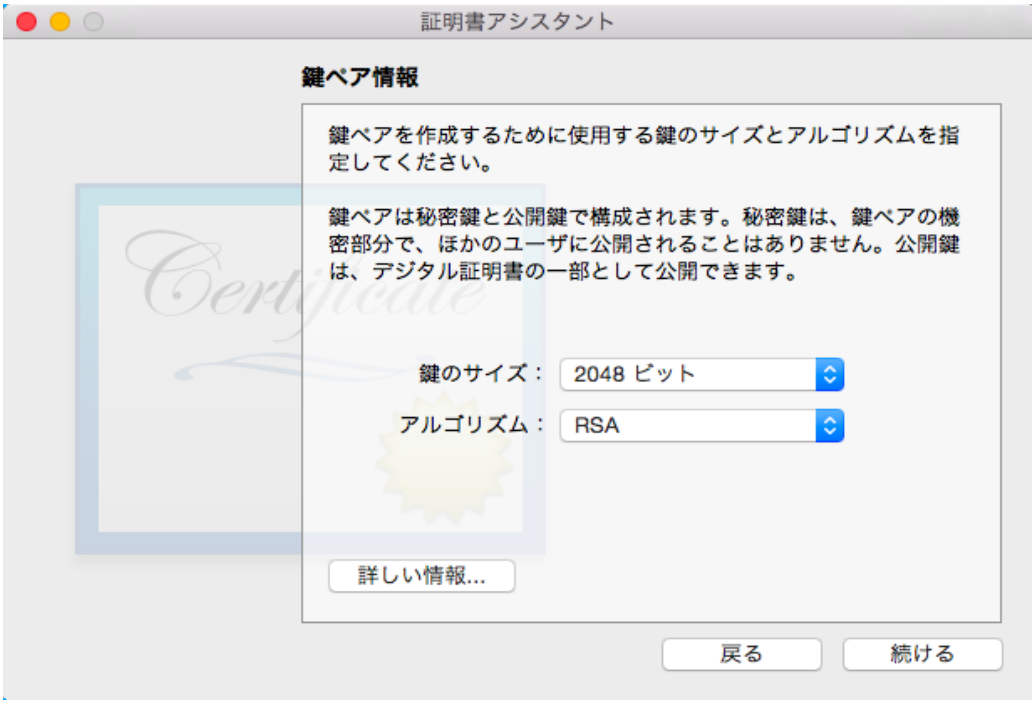

図　2- 12 鍵ペア情報

2.6 iOS Developer Program

Apple 社が Mac OS X，iOS 開発者向けに提供している，サポートサービスである．iOS Developer Program に加入すると, iOS の開発版や最新版をダウンロード･使用出来る上に，App Store にアプリケーションソフトをリリースすることも出来る．また，テクニカルサポートや開発者フォーラムを利用出来る．実機での動作検証やアドホック配信（実機検証協力者向けの小規模配信・最大 100 デバイス）の実施，“そして App Store への公開申請には iOS Developer Program への加入が必要．そしてアップルは App Store 経由以外での iPhone アプリケーションの販売を認めていないため，有償・無償を問わず公式に iPhone アプリケーションの販売（配布）を検討している開発者は必ず加入しなければならない”[8].

2.7 iOS Developer Program から証明書ファイルを作成

iOS Developer Program から証明書を作成する．

① iOS Dev Center へアクセスし「Certificates, Identifiers & Profiles」を選択する.

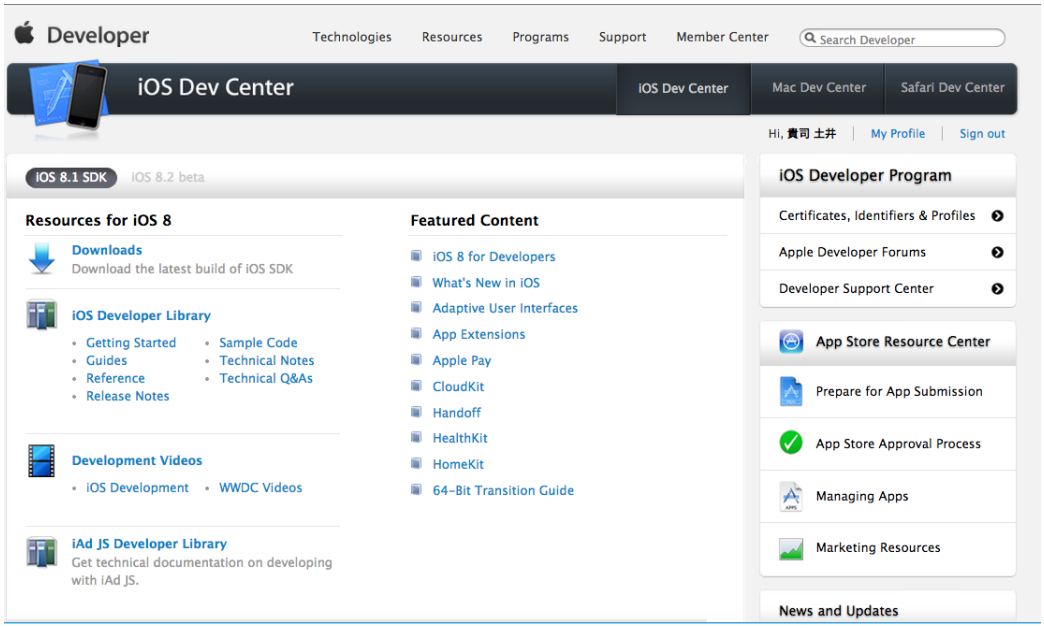

図 2- 13 iOS Dev Center

② Certificates を選択する.

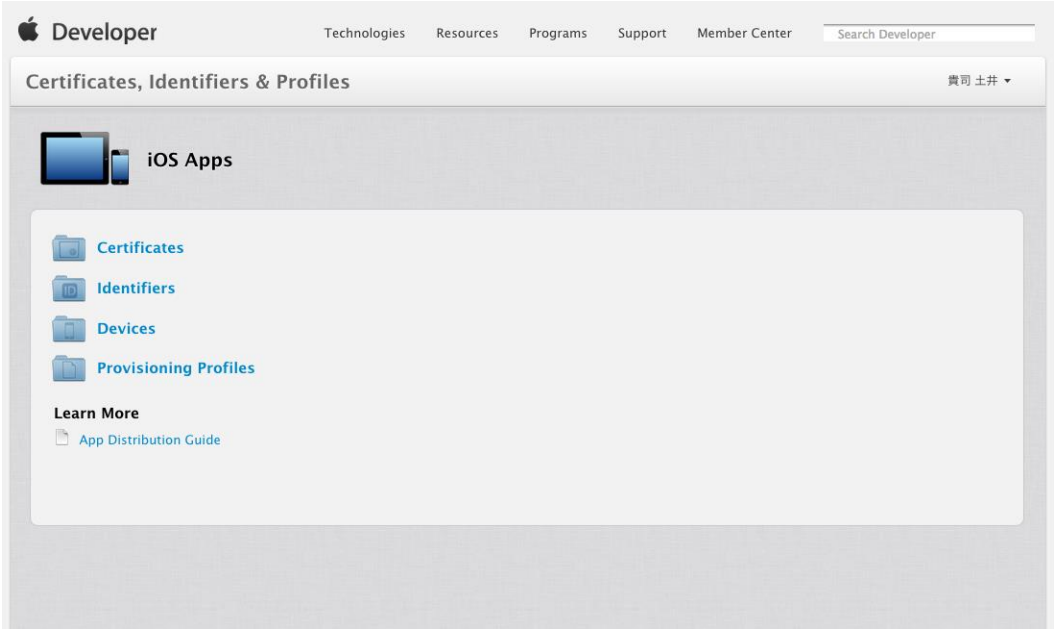

図 2- 14 Certificates, Identifiers & Profiles

③　左メニューの Certificate を選択して「+」ボタンを押下する.

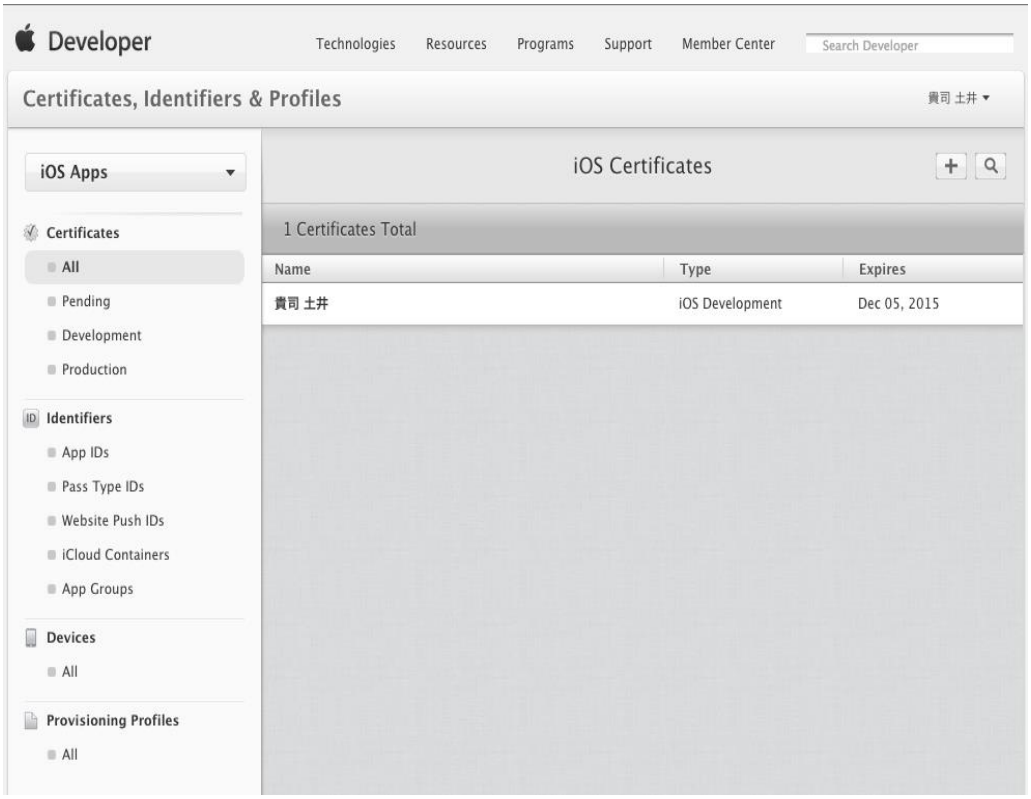

図 2- 15 証明書ファイル追加

④　証明書の種類は以下の種類がある．

- Development：開発用
  - iOS App Development：通常時
  - pple Push Notification service SSL (Sandbox)：通知を利用する場合

- Production：本番用
  - App Store and Ad Hoc：通常時
  - Apple Push Notification service SSL (Production)：通知を利用する場合
  - Pass Type ID Certificate：PassBook 用
  - Website Push ID Certificate：Safari 通知用

開発と本番や Push 通知を使う場合によって選択する内容が異なる．今回は，開発用かつ通常時となる[6].

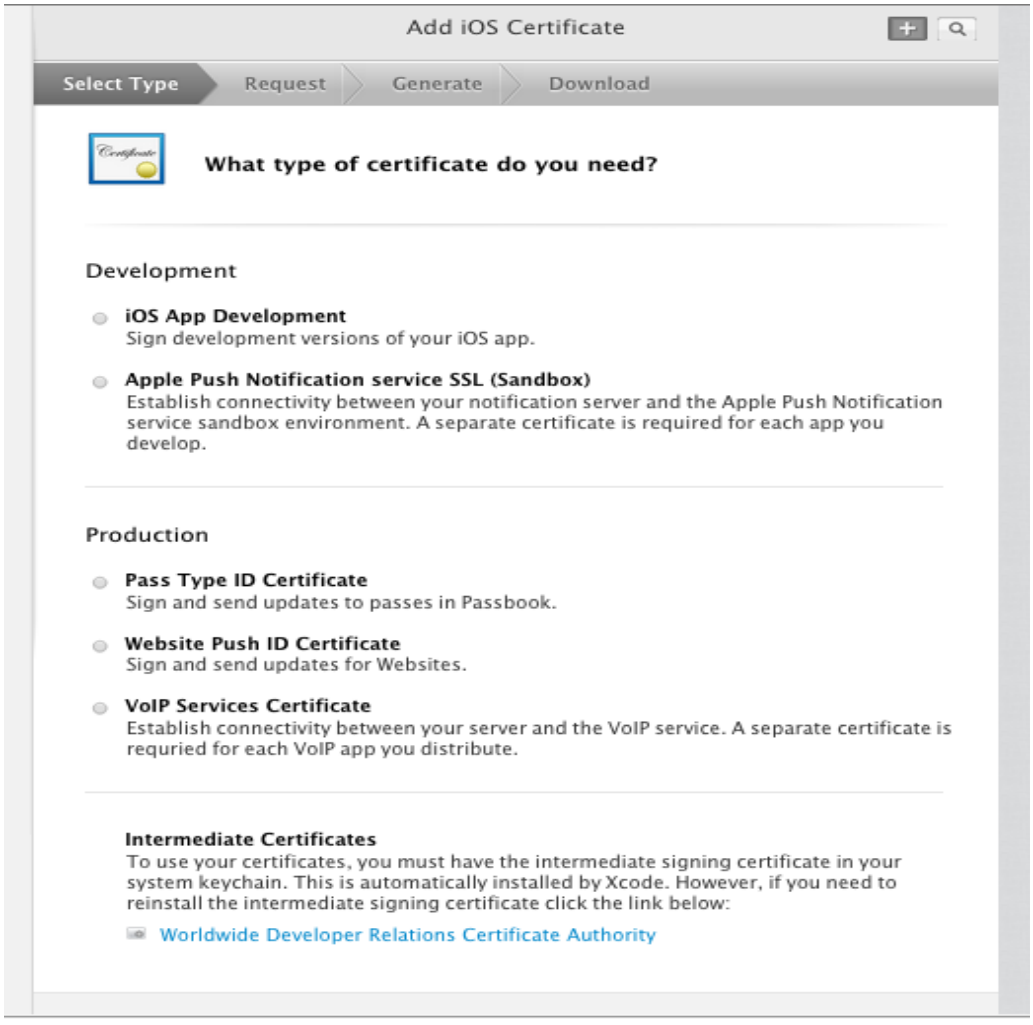

図 2- 16 証明書種類の選択

⑤　問題ない場合「Continue」を押下する.

図 2- 17 証明書作成要求(CSR)の確認

⑥　「Choose File...」ボタンを押下し，先ほど作成した証明書作成要求(CSR)を選択し，証明書作成要求(CSR)が選択されたのちに，「Generate」ボタンを押下する.

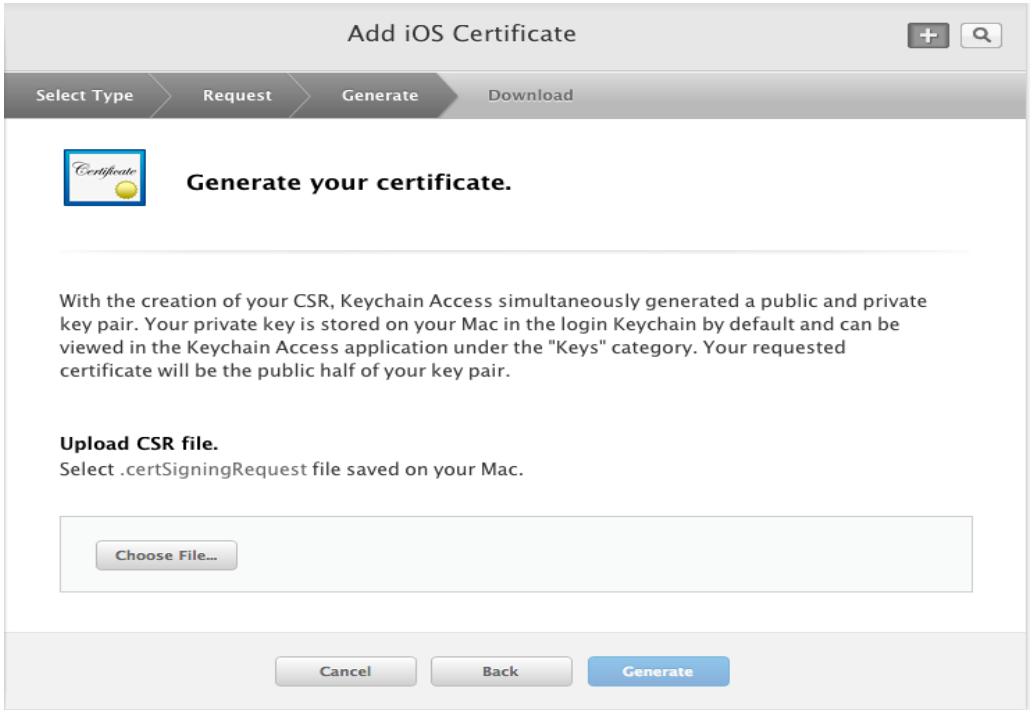

図 2- 18 証明書作成要求(CSR)の選択

⑦　証明書ファイルが作成されたことを確認できたのちに, 作成された開発用証明書(CER)のダウンロードを行う.

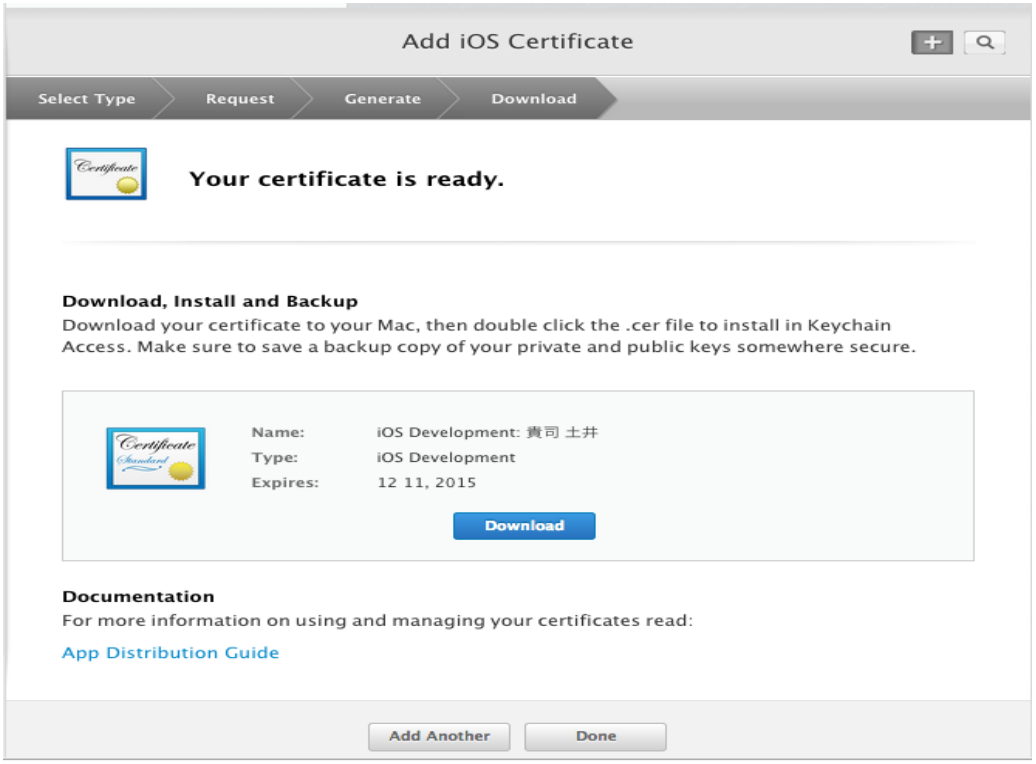

図 2- 19 開発用証明書(CER)のダウンロード

2.8 Mac に開発用証明書(CER)を登録

Mac に開発用証明書(CER)の登録を行う．ダウンロードした，保存先にある「ios_development.cer 」ファイルをダブルクリックすると，追加ボタンがあるので，追加ボタンを押下する．しかし，ファイルをダブルクリックしたのちに「“システムルート”キーチェーンは変更できません.」と，表示された場合は，キーチェーンアクセスのメニューのファイルから読み込むを選択し，開発用証明書(CER)である「ios_development.cer 」ファイルを選択すると読み込める．

① 読み込まれた「ios_development.cer 」ファイルは，ログインキーチェーンに登録される．左のメニュー最下段の「証明書」を押下し，登録されている事を確認する．ここに出てくる開発者名は，Apple ID として登録している名前である．

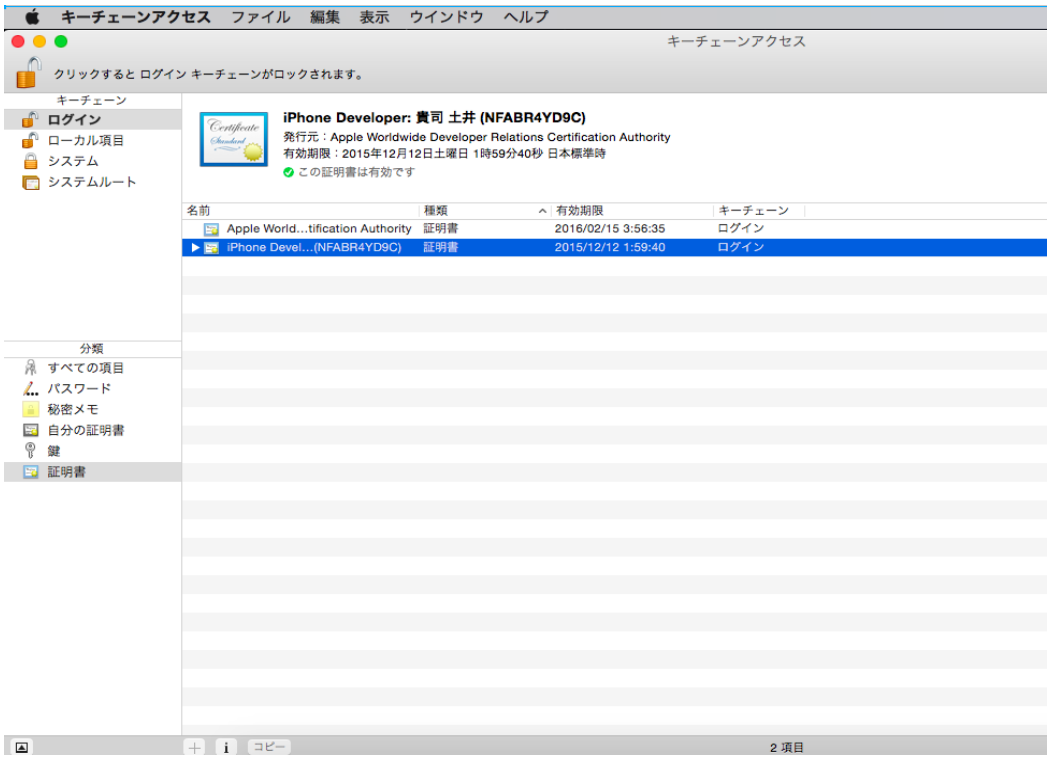

図 2- 20 キーチェーンアクセスでの証明書登録

② 左のメニューの下から 2 番目の「鍵」を選び，秘密鍵と公開鍵が登録されている事を確認する．これで証明の登録が完了．

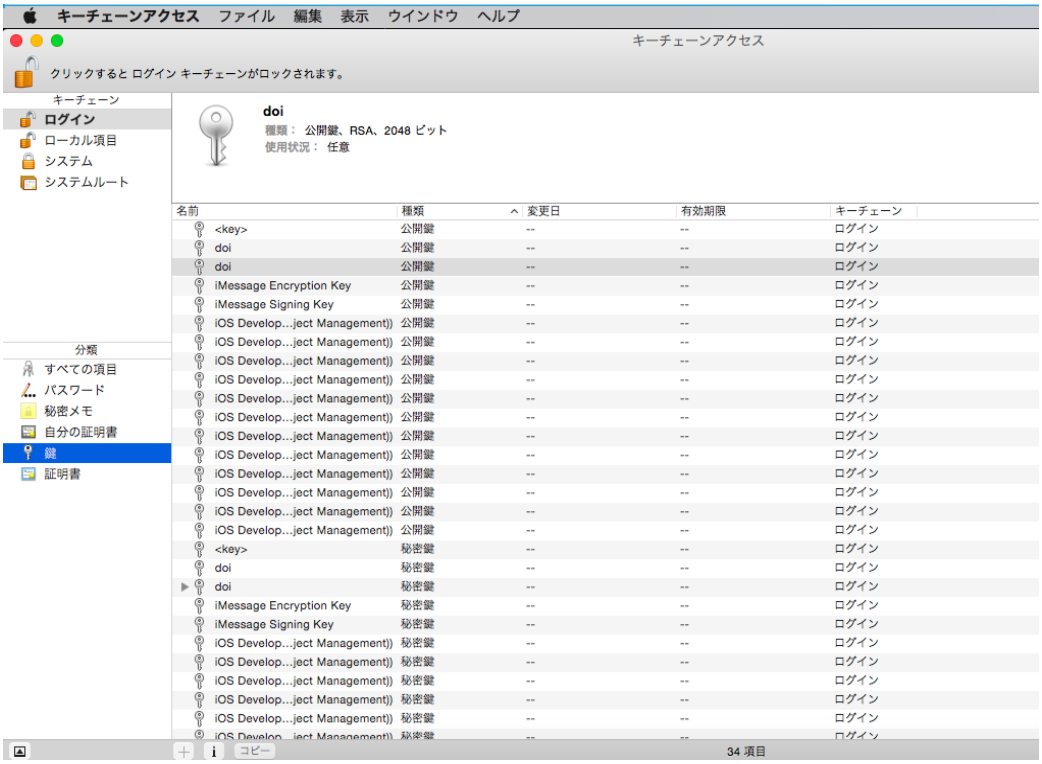

図 2- 21 秘密鍵と公開鍵

2.9 App ID の登録

App ID はそれぞれの iOS アプリケーションの ID を指している．“App ID は，キーチェーンのアクセスのために用いるための識別子となる．この識別子も，Bundle Identifire と同様の命名規則が推奨されている．App ID は，後の工程でプロビジョニングファイルに埋め込まれ，適切な開発者によってアプリケーションがインストールされているのかの認証として機能する”[9].

① 「Certificates, Identifiers & Profiles」から左のメニューの「Identifiers」の「App IDs」を選択する．

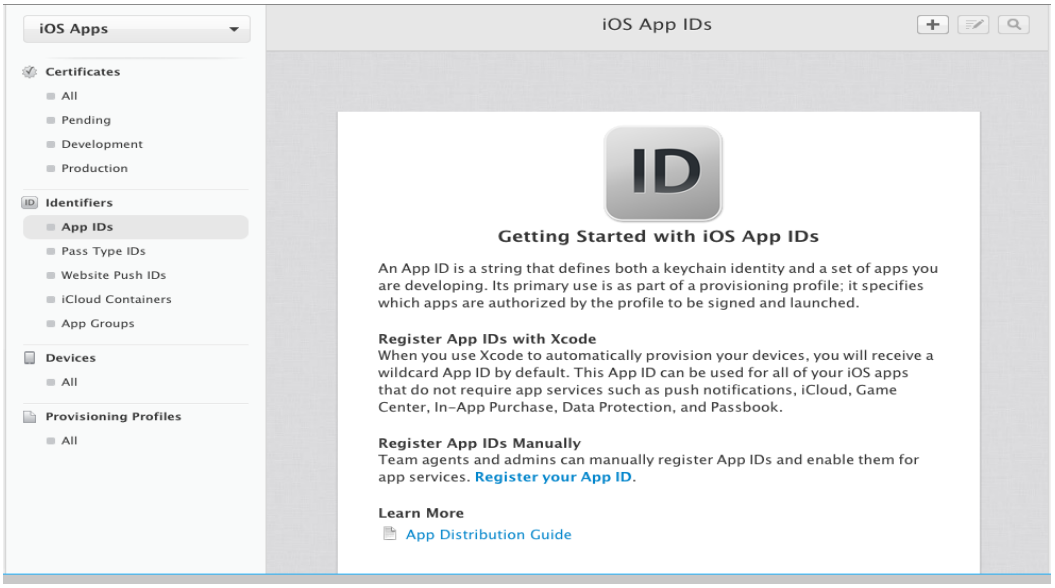

図 2- 22 Certificates, Identifiers & Profiles

② 「＋」（新規作成）を選択する．

図 2- 23 iOS App IDs

③　App ID の作成フォームが表示される

- App ID Description
  App ID の名前を入力する．このとき，アプリ用の ID として他と識別できるような名前を付ける．iTunes Connect でアプリ情報を設定するときに表示される名前になる．

- App ID Prefix
  同じ App ID でも一意に判別できるように Prefix を指定する．チームの ID を指定する．

- App ID Suffix
  - Explicit App ID：指定された Bundle ID のみ有効な App ID を作成する．アプリごとに App ID が必要な場合はこちらを指定する．
  - Wildcard App ID：Bundle ID を指定しないため，App ID を使い回すことができる．ただし，いくつか利用できるサービスに制限が加わるため注意が必要である．

  ※ここでは, Wildcard App ID にする. これは, 任意の名前を付ける事ができるが, App Store などで配布する際には，必ず他人と重ならないようにする．そのため, 慣例的に自分のサイトの URL をひっくり返した名前が使われる. また, 最後に「.*」を入れる事により，複数のアプリに共通の App ID を使う事ができる．
  例：jp.takashi.doi.*

- App Services
  アプリ内で使用しているサービスを設定する．開発したアプリで使っているサービスにチェックを付ける．Game Center は必ず有効になる．

すべての設定が完了したら「Continue」をクリックし作成する.

図 2- 24 iOS App ID の作成

④ 確認できたら，「Submit」を押下する．

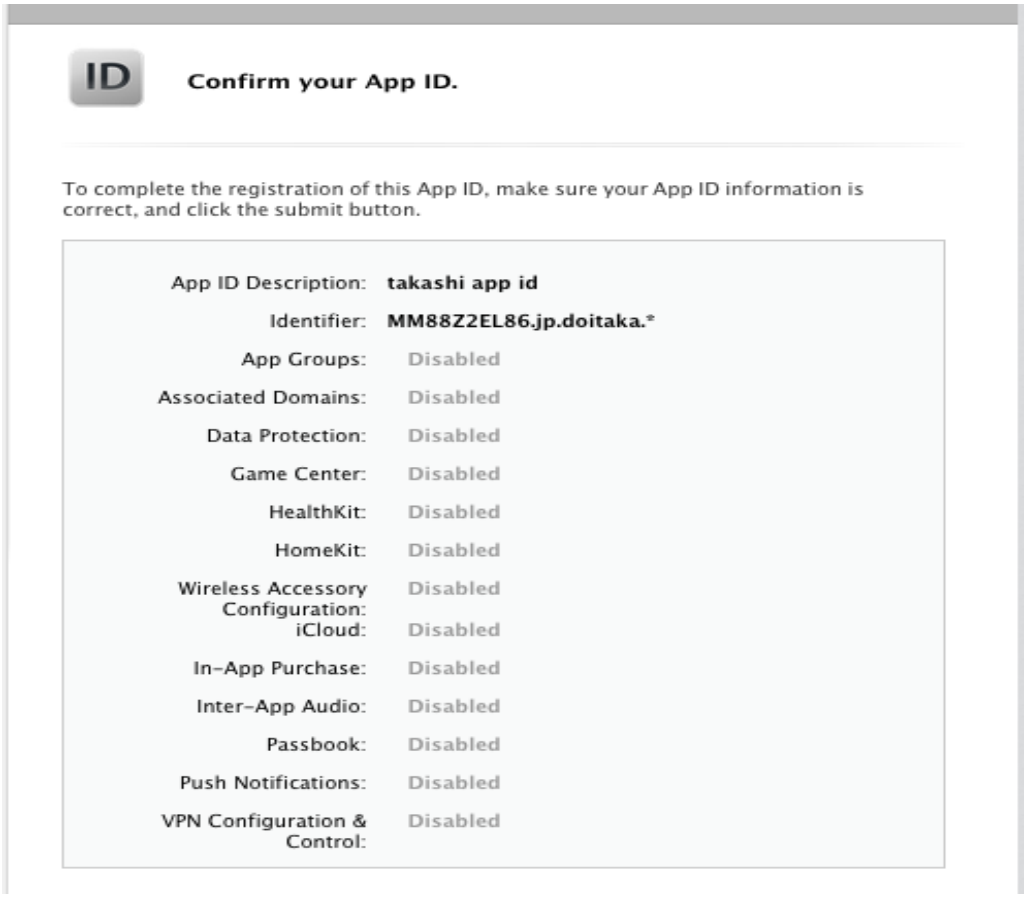

図 2- 25 登録確認

2.10 デバイス登録

① デバイスも同じように左のメニューから「All」を選択し「＋」（新規作成）を選択する. この時, デバイス名は任意だが, 分かりやすい名前を付ける方がよい. 更に「UDID」という，それぞれのデバイス固有の識別子を登録する

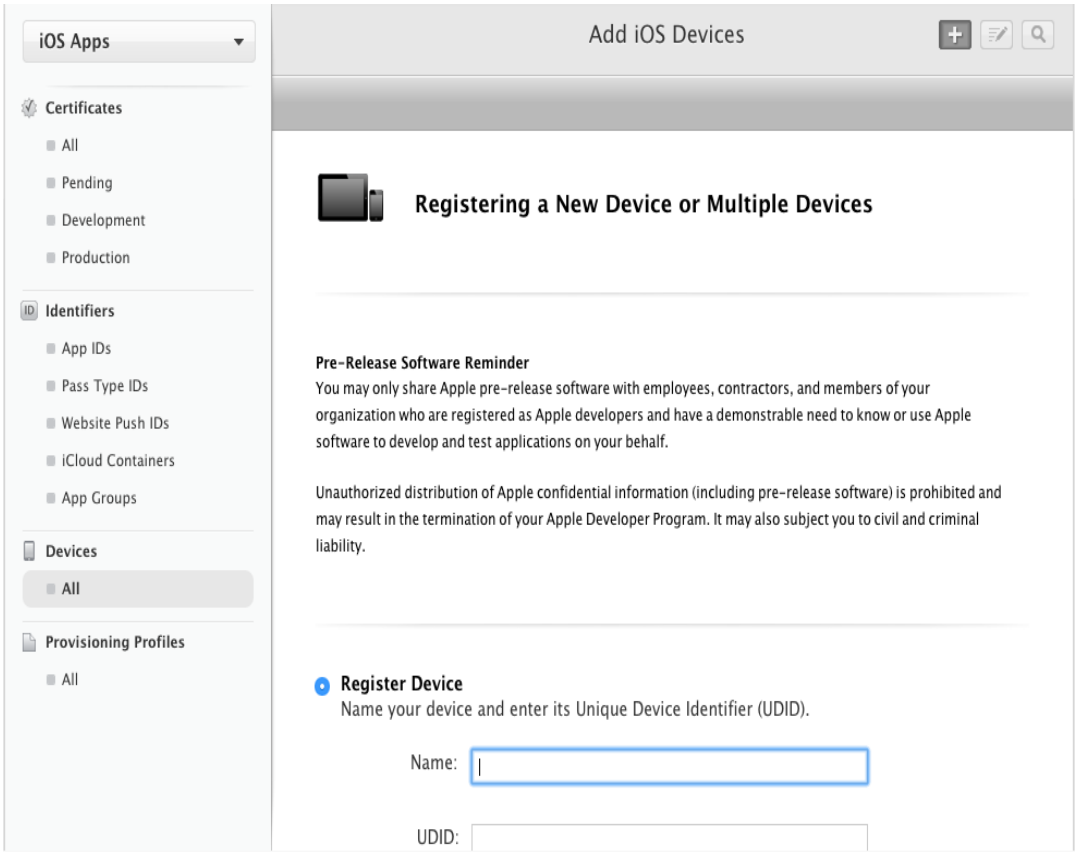

図 2- 26 デバイス登録

② この UDID を調べるには，アプリケーションから XCode を開き，さらに XCode の Window メニューから Devices を起動する．そして，「Identifier」の欄に書かれている半角英数字の識別子をウェブページの「UDID」の欄に記入し，「Continue」を押下する．

図 2- 27 UDID 確認

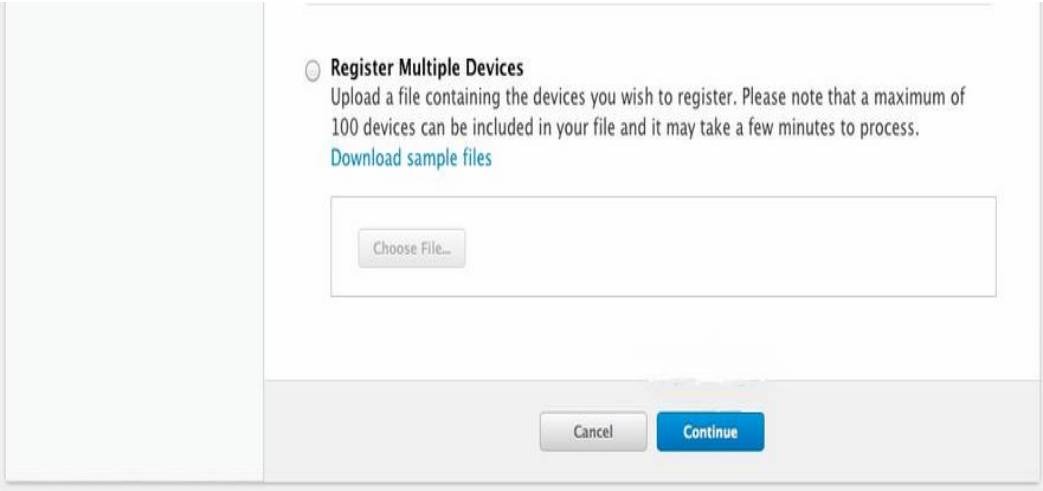

図 2- 28 デバイス登録

③　登録内容を確認する.

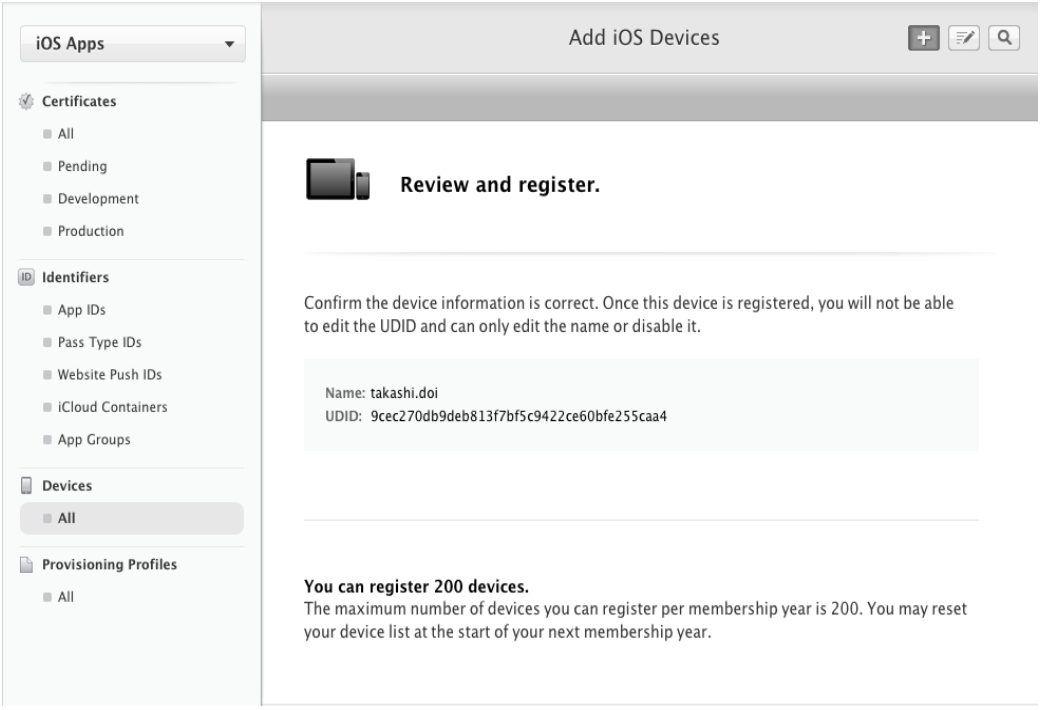

図 2- 29 登録確認

④　「Done」を押下し，登録が完了する.

図 2- 30 登録完了

2.11 プロビジョニングファイル

プロビジョニングプロファイルは，アプリケーションを iOS デバイス上で動作可能にする為に必要なリソースです．これが無いと，App Store でアプリケーションを配布することはもちろん，自分の iPhone に開発中のアプリケーションをインストールして，実行することも出来ません．プロビジョニングプロファイルは，App ID，証明書，テスト端末の UDID の３つの情報を関連づけるものです．これらの関連付けによって，アプリケーションの識別や，その妥当性，またテストの為にアプリケーションをインストールできる端末を指定するといったことが実現されています．プロビジョニングプロファイルは，Apple 社のホストする Web サイトである「iOS Dev Center」で作成・管理します．プロビジョニングプロファイルには，開発用，アドホック用（テスト等の為に配布する），App Store 用と３種類があります．

- 開発用
  - [役割] : 開発チームがアプリケーションをテストするために利用するために必要
  - [内容] : App ID，アプリをビルド可能な開発者の証明書（複数），インストールできるデバイスの UDID（複数）

- AdHoc 用
  - [役割] : 外部のテスト担当者が特定のデバイス上でアプリケーションを実行するために必要
  - [内容] : App ID，配布用証明書（１つ），インストールできるデバイスの UDID のリスト

- App Store 用
  - [役割] : App Store でアプリケーションを配布するために必要
  - [内容] App ID、配布用の証明書（１つ）

それぞれ役割が異なる．また役割ごとに必要なリソースがパッケージされていることに注意してください[10].

① Provisioning Profile を選択し，「+」（新規作成）を選択する．

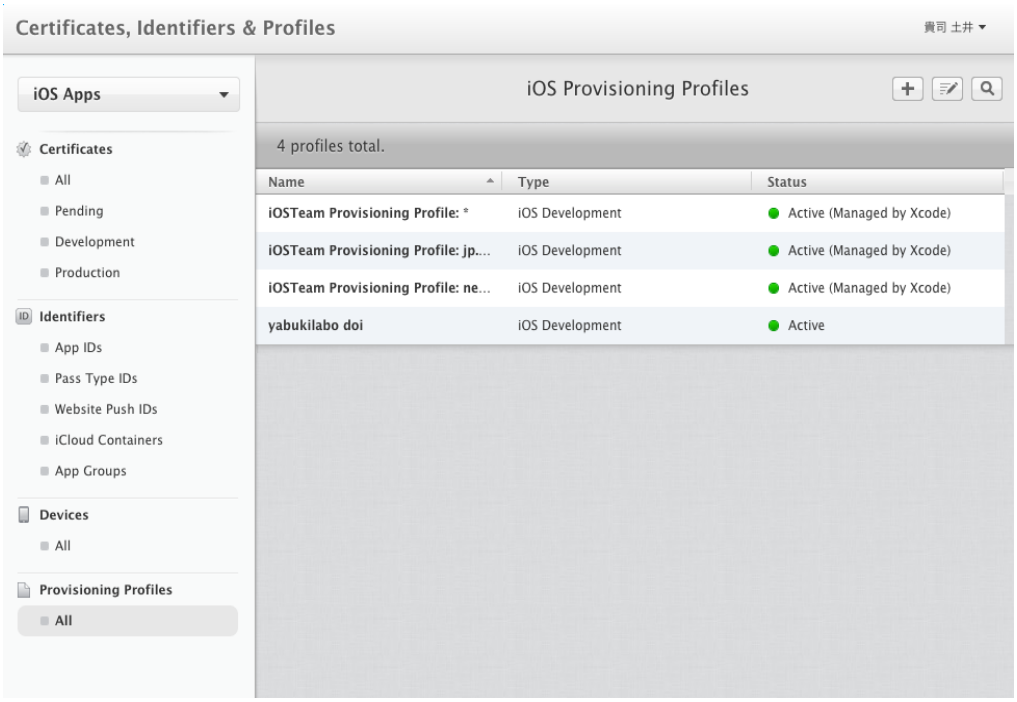

図 2- 31 プロビジョニングファイル

② 作成するプロビジョニングプロファイルのタイプを選択する．ここでは，iOS App Development を選択する．

- Development：開発用であり，開発に利用するプロビジョニングプロファイルを作成する場合

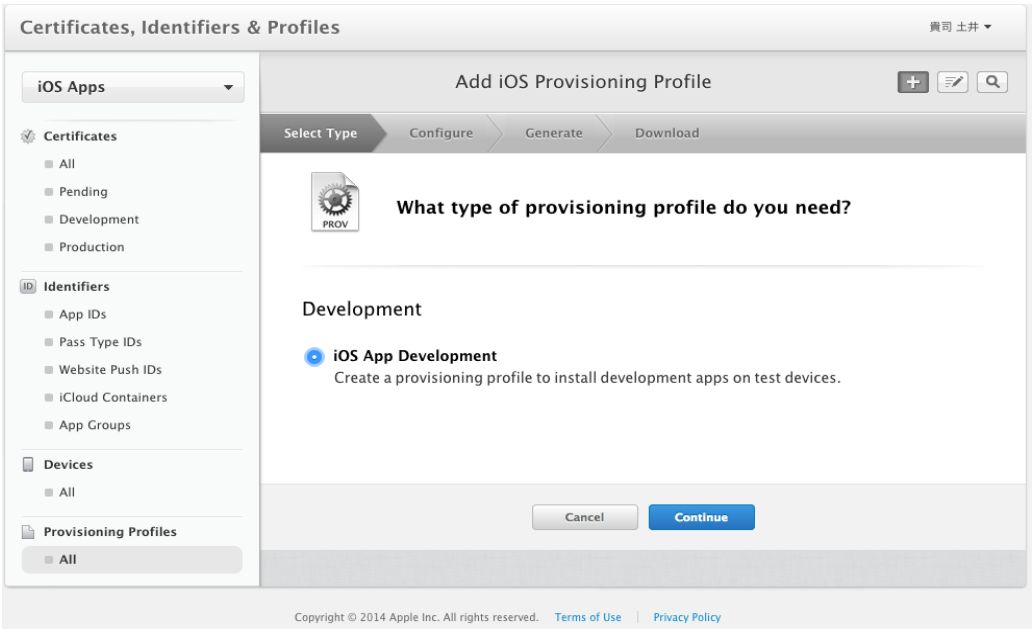

図 2- 32 Development 選択

③　App ID を使うか問われるので，先ほど作った App ID を選択し，「Continue」を選択する．

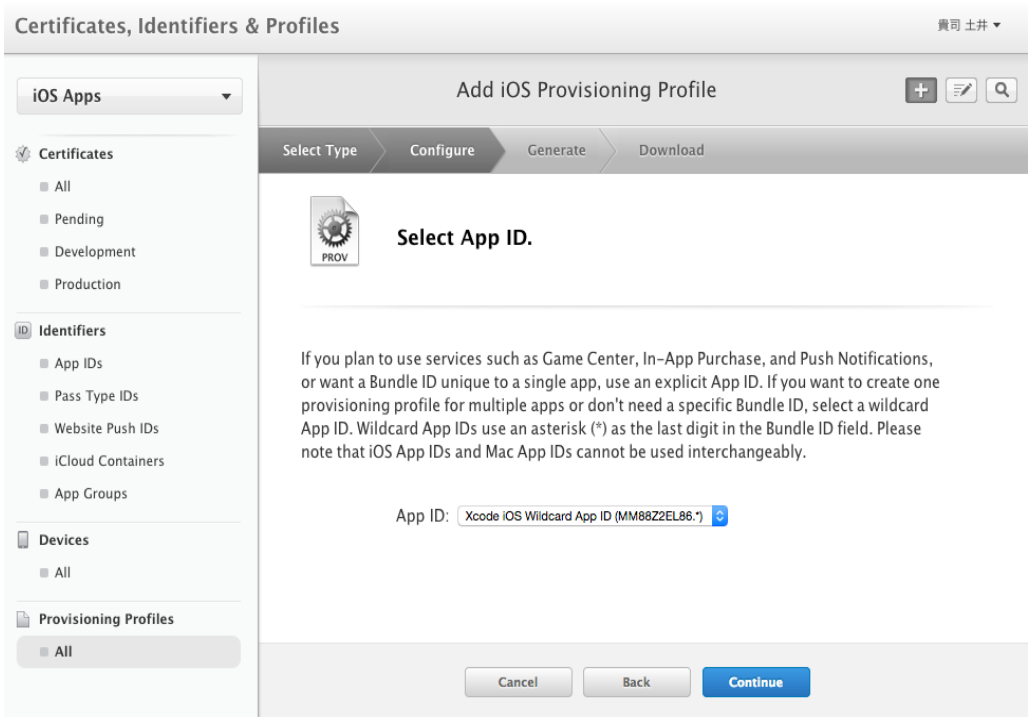

図　2- 33 Select App ID 選択

④　どの Certificate を使うか問われるので，登録した Certificate を選択し，「Continue」を選択する．

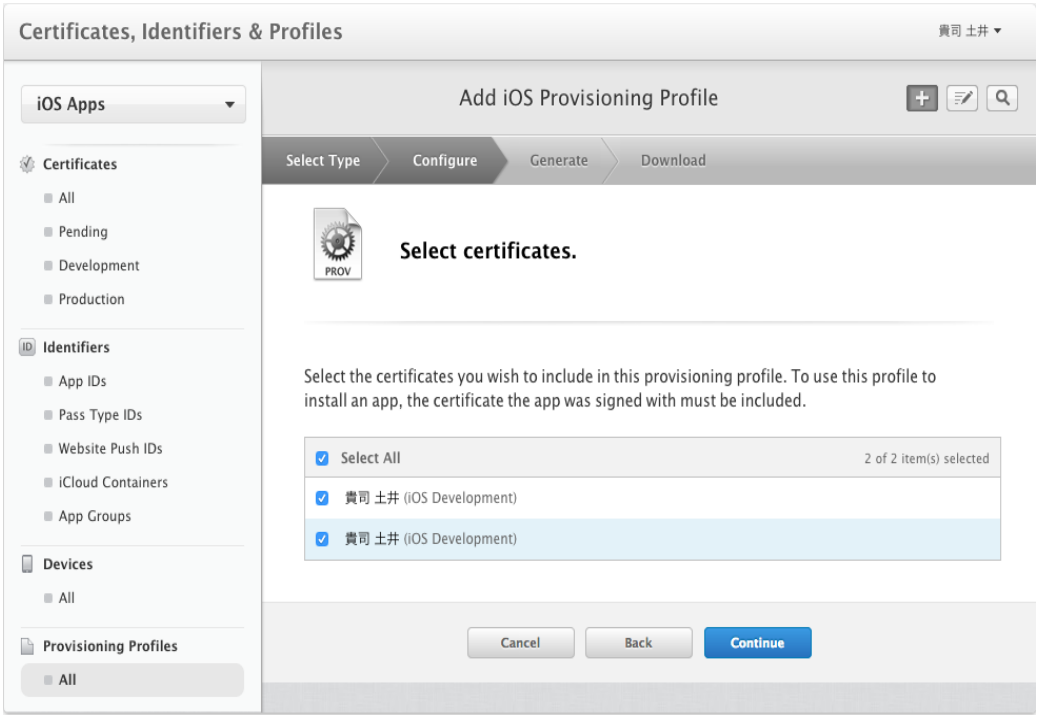

図　2- 34 Select certificates

⑤　アプリケーションを実行する対象の端末を指定し，「Continue」を選択する．

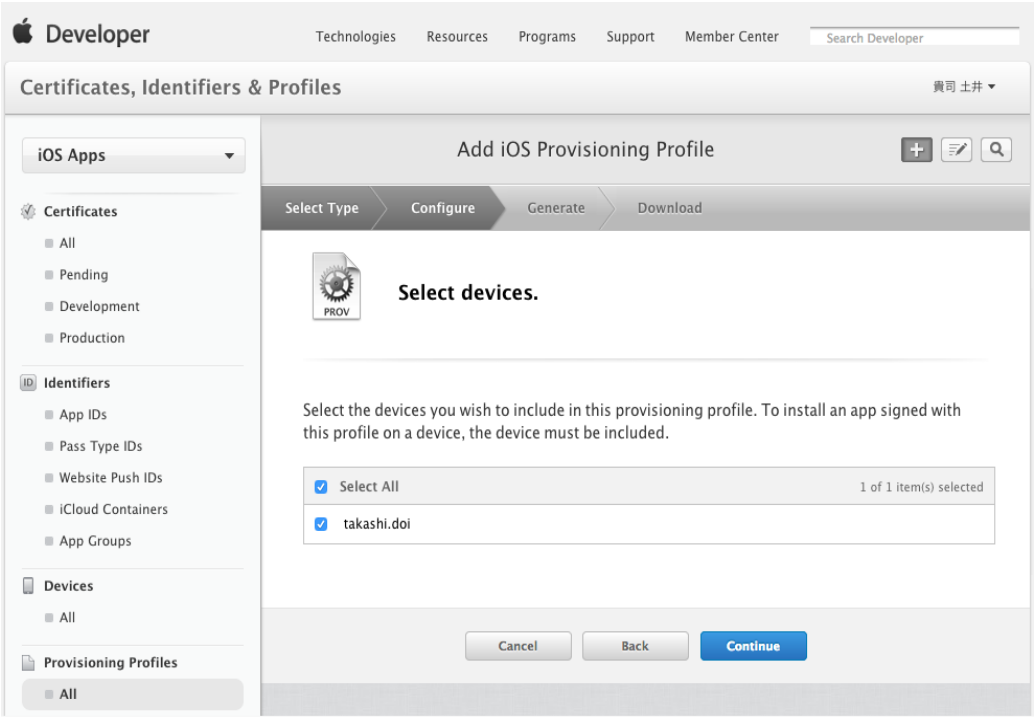

図　2- 35 Select devices

⑥　プロビジョニングプロファイルのプロファイル名を設定し，「Generate」を選択する．

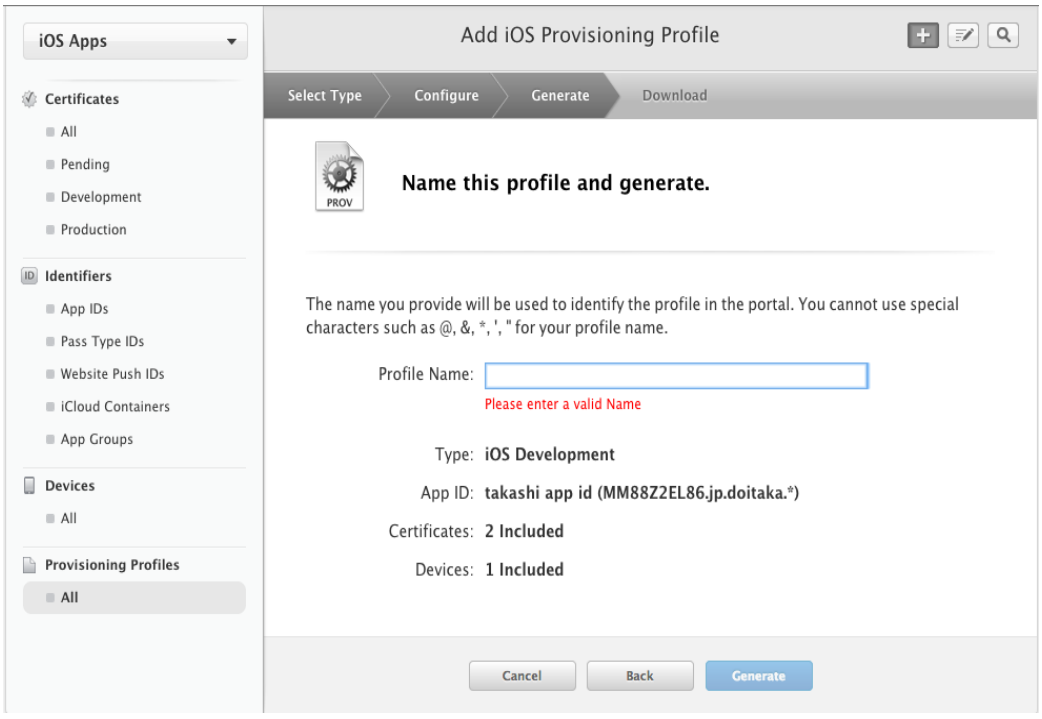

図　2- 36 プロファイル名

⑦　作成完了画面になり，ダウンロードを行う．

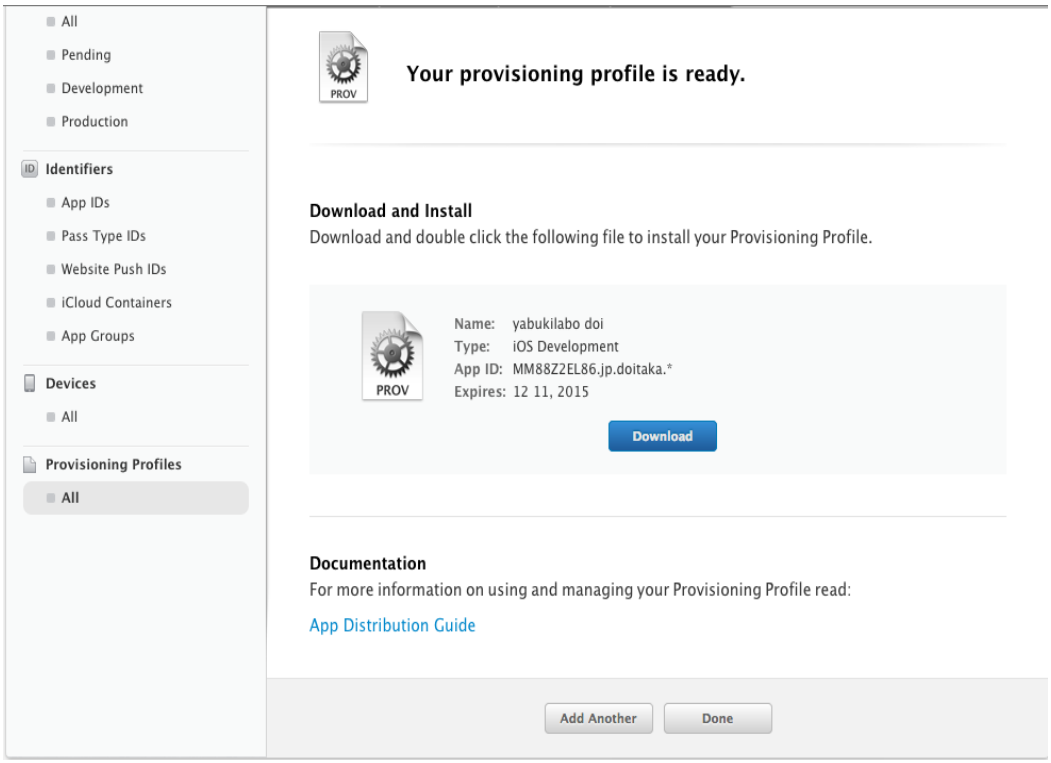

図 2- 37 ダウンロード

2.12 Xcode にプロビジョニングプロファイルを設定する

Xcode の「Build Settings」タブを開き，Code Signing の下にある[Provisioning Profile]メニューから該当するプロビジョニングプロファイルを指定する．

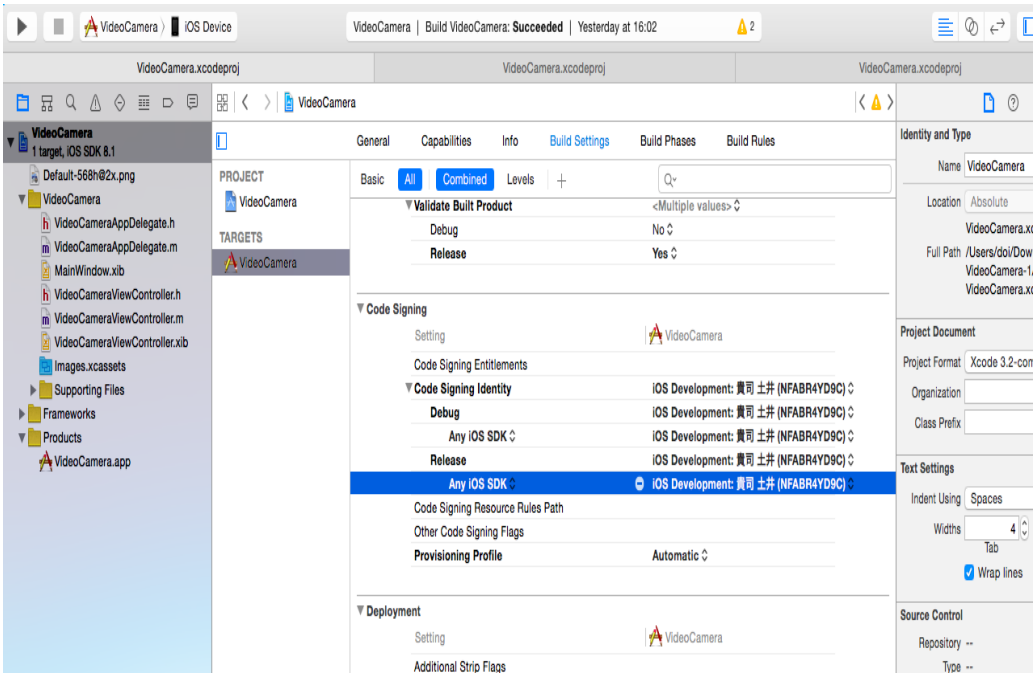

図 2- 38 プロビジョニングプロファイル設定

2.13 iOS アプリのプロビジョニング図

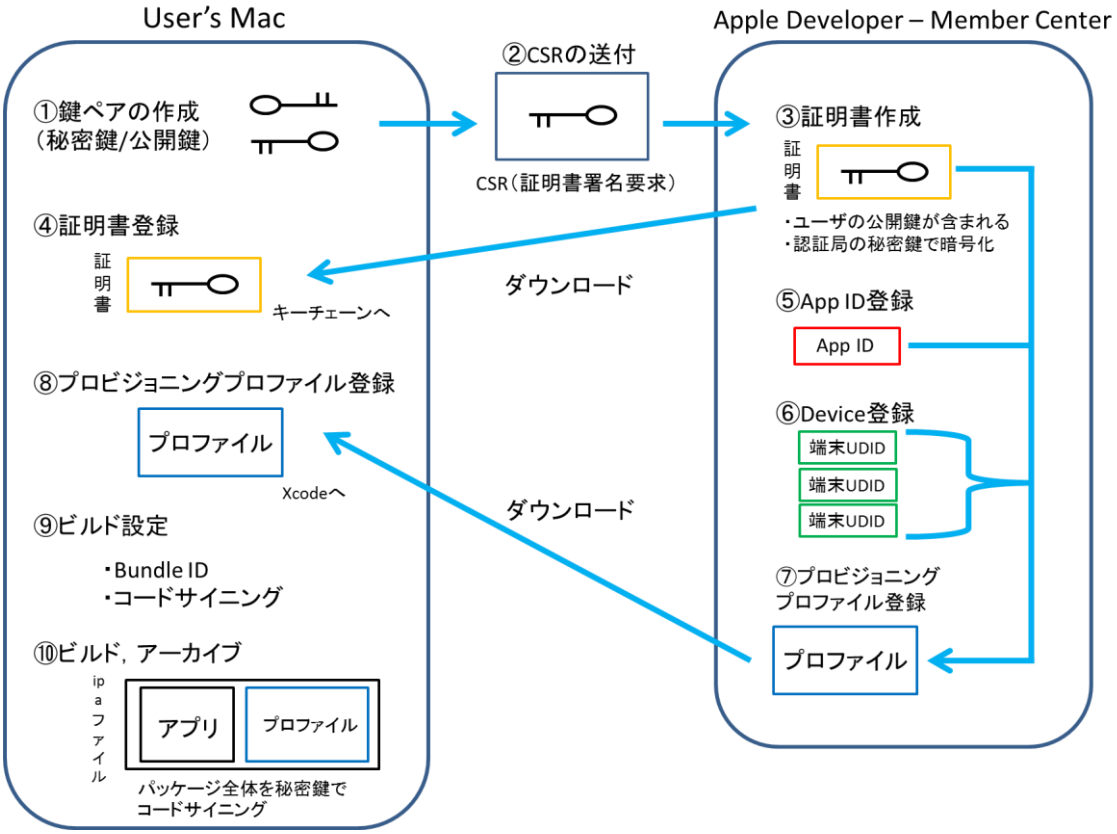

図 2- 39 概念図 引用文献[11]

1. 鍵ペア（秘密鍵／公開鍵）を作成.
   ※秘密鍵は端末内から外へは出ない.
2. 公開鍵を CSR（証明書署名要求）として認証局（Apple）に送付.
3. 証明書発行
   ※受け取った公開鍵から証明書を作成し，認証局の秘密鍵で署名.
4. 発行された証明書を DL してキーチェーンに登録.
5. App ID を登録.
6. 実行を許可する端末を登録.
   ※端末の UDID（複数）.
7. プロビジョニングファイルを作成
   ※証明書，App ID，実行許可端末を関連付け.
8. プロビジョニングファイルを DL して端末内に登録.
9. 配布したいアプリケーションに BundleID，コードサイニング（署名）の設定を行う.
10. アプリケーションをビルド，アーカイブし，パッケージファイル（ipa ファイル）を作成.
    ※ipa ファイルは ZIP 形式でまとめられており，プロビジョニングファイルを含む.
    ※パッケージファイル全体に署名を施す[11].

参考文献

[3] iOS. [Apple. iOS]. http://e-words.jp/w/iOS-2.html, (参照 2014-10-17)
[4] Xcode. http://e-words.jp/w/Xcode.html, (参照 2014-10-17)
[5] CSR. [Certificate Signing Request]. 証明書署名要求. http://e-words.jp/w/CSR.html, (参照 2014-10-17)
[6] よく分かる！iOS アプリ開発に必要な証明書ファイルの作成方法.
http://dev.classmethod.jp/smartphone/ios-certificates/, (参照 2014-12-12)
[7] Apple Developer Connection. http://ja.wikipedia.org/wiki/Apple_Developer_Connection,
[8] It_lives_vainly の日記. [iPhone] App ID's とアプリケーションのインストールについて.
http://d.hatena.ne.jp/It_lives_vainly/20090204/1233741543, (参照 2014-12-12)
[9].よく分かる！iOS アプリ開発に必要な証明書ファイルの作成方法 | アドカレ 2013 : SP. #4. http://dev.classmethod.jp/smartphone/ios-certificates/, (参照 2014-12-12)
[10] iOS プロビジョニングプロファイルの作成. http://foreignkey.jp/archives/1414,
[11] iOS アプリのプロビジョニング周りを図にしてみる.
http://qiita.com/fujisan3/items/d037e3c40a0acc46f618,

第 3 章　iOS 開発

3.1 本章の構成

本章では，前章で登録を行った Xcode を用い，リアルタイムに動画を処理するスマートフォンアプリケーションの開発をする．本研究で目的とする，リアルタイムに動画を処理する（動くものを消す）プログラムの解説，それぞれに必要なソースコードの記述と解説を行う．

3.2 メディアキャプチャ

使うのは、メディアキャプチャと呼ばれる機能である．メディアキャプチャは，iOS4 から導入された機能で，AVFoundation フレームワークが提供する．前が「AVCapture」で始まるクラス群がそれにあたる．オーディオやビデオといったメディアデータを，直接取り扱う事のできる機能だ．メディアキャプチャと呼ばれる機能である．中心となるのは，セッションと呼ばれる概念だ．メディアのキャプチャは，セッションを作る事から始まる．セッションを表すクラスは，AVCaptureSession だ．キャプチャを行うには，セッションにインプットとアウトプットをつなげてやる．インプットは，カメラやマイクといったデバイスからの入力となる．カメラならばビデオメディアの入力，マイクならばオーディオメディアの入力となるだろう．アウトプットは，メディアデータの書き出し先となる．ファイルに書き込む出力もあれば，生のデータをバッファを経由して受け取ってプログラム中で処理するという出力もある．インプットを表すクラスは AVCaptureInput，アウトプットのクラスは AVCaptureOutput となる．それぞれ，デバイスやメディアに応じて，様々なサブクラスが用意されている．セッション，インプット，アウトプットの関係を図で表すと，次のようになる．

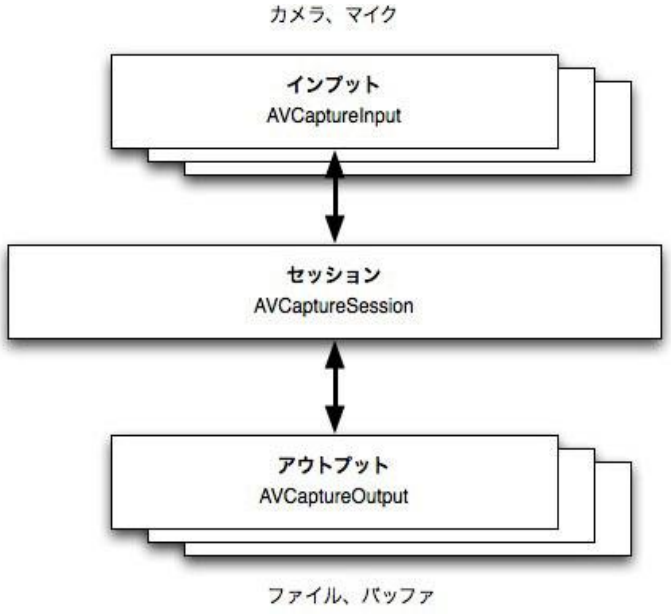

図 3- 1 メディアキャプチャ図　引用文献 [12]

3.3 セッションの作成

インプットとしてビデオカメラデバイス，アウトプットとしてビデオデータを直接受け取るものを指定しよう．ソースコードを書くために Xcode で新たなプロジェクトを作る事になると思うが，そのときに使用するフレームワークを追加する事になる。以下のものを追加してほしい．

- AVFoundation.framework
- CoreVideo.framework
- CoreMedia.framework

また，キャプチャに関する入力と出力の管理を行う，AVCaptureSession 型の変数をクラスのインスタンス変数として用意する．

Xcode のナビゲーターエリアから, TARGETS の General を選択すると, Linked Frameworks and Libraries という項目があるので，そこから選択する．

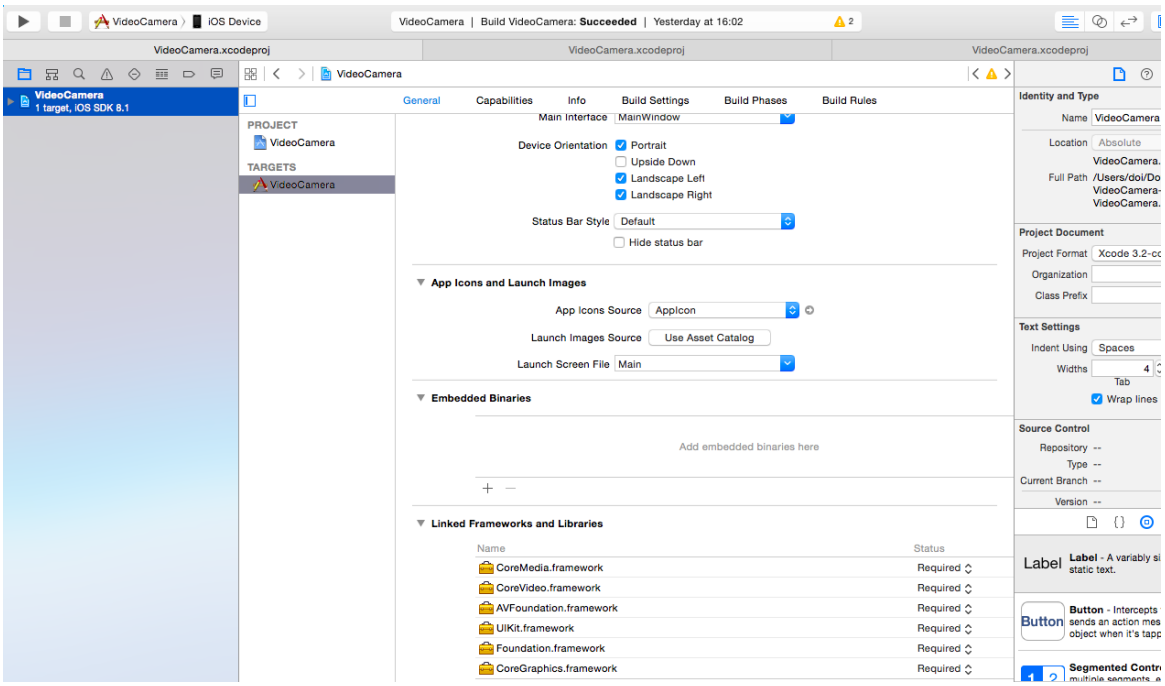

図 3- 2 ナビゲーターエリア

- ソースコード

AppDelegate.h

```
#import <UIKit/UIKit.h>

@class VideoCameraViewController;


@interface VideoCameraAppDelegate : NSObject <UIApplicationDelegate> {

}


@property (nonatomic, retain) IBOutlet UIWindow *window;


@property (nonatomic, retain) IBOutlet VideoCameraViewController *viewController;


@end
```

AppDelegate.m

```
#import "VideoCameraAppDelegate.h"

#import "VideoCameraViewController.h"

@implementation VideoCameraAppDelegate

@synthesize window=_window;

@synthesize viewController=_viewController;

- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions{
      self.window.rootViewController = self.viewController;
      [self.window makeKeyAndVisible];
      return YES;
}

- (void)applicationWillResignActive:(UIApplication *)application{
}

- (void)applicationDidEnterBackground:(UIApplication *)application{
}

- (void)applicationWillEnterForeground:(UIApplication *)application{
}

- (void)applicationDidBecomeActive:(UIApplication *)application{
}

- (void)applicationWillTerminate:(UIApplication *)application{
}
```

続き

```
- (void)dealloc{
	[_window release];
	[_viewController release];
	[super dealloc];
}

@end
```

ViewController.h

```
#import <UIKit/UIKit.h>
#import <AVFoundation/AVFoundation.h>

@interface VideoCameraViewController : UIViewController
{
①	AVCaptureSession*	_session;
}

@end
```

① 作成した AVCaptureSession クラスは，ここに代入する．

- ソースコード

ViewController.m

```
#import “VideoCameraViewController.h”

@implementation VideoCameraViewController

- (void)viewDidLoad
{
    // ビデオキャプチャデバイスの取得
②   AVCaptureDevice*       device;
    device = [AVCaptureDevice defaultDeviceWithMediaType:AVMediaTypeVideo];

    // デバイス入力の取得
③   AVCaptureDeviceInput*     deviceInput;
    deviceInput = [AVCaptureDeviceInput deviceInputWithDevice:device error:NULL];

    // ビデオデータ出力の作成
    NSMutableDictionary*            settings;
④   AVCaptureVideoDataOutput*     dataOutput;
    settings = [NSMutableDictionary dictionary];
    [settings setObject:[NSNumber numberWithInt:kCVPixelFormatType_32BGRA]
                forKey:(id)kCVPixelBufferPixelFormatTypeKey];
    dataOutput = [[AVCaptureVideoDataOutput alloc] init];
    [dataOutput autorelease];
    dataOutput.videoSettings = settings;
    [dataOutput setSampleBufferDelegate:self queue:dispatch_get_main_queue()];

    // セッションの作成
⑤   _session = [[AVCaptureSession alloc] init];
    [_session addInput:deviceInput];
    [_session addOutput:dataOutput];

    // セッションの開始
    [_session startRunning];
}
```

② ビデオカメラデバイスを取得する.これには，AVCaptureDevice というクラスを使う．このクラスが提供する defaultDeviceWithMediaType:というメソッドを使えば，指定したメディアに対するデバイスを取得する事ができる．このメソッドはデフォルトデバイスを取得するので，返ってくるデバイスはただ 1 つに決まる．iPhone4 のように表と裏で 2 つのカメラを持っている場合，その両方を取得するためのデバイスも別途用意されている．

③ インプットを作成する．ここでは AVCaptureDeviceInput というサブクラスを使う．先ほど取得した AVCaptureDevice を指定して，インスタンスを作成する．

④ アウトプットの作成する．AVCaptureVideoDataOutput というクラスを使う．ビデオデータを生のまま受け取るために使うクラスだ．設定として，ピクセルフォーットを指定してやる．

⑤ セッションを作成する．AVCaptureSession をインスタンス化する．そして，addInput:メソッドを使ってインプットを追加, addOutput:メソッドでアウトプットを追加する．

これで準備完了だ[13].

3.4 AVCaptureVideoDataOutput のデリケート

AVCaptureVideoDataOutput クラスは，キャプチャしたデータをそのままアプリに渡してくれるものだ．そのデータを受け取るために，デリゲートメソッドが用意されている．AVCaptureVideoDataOutputSampleBufferDelegate プロトコルで定義される，captureOutput:didOutputSampleBuffer:fromConnection:メソッドだ．

- ソースコード

ViewController.m

```
- (void)captureOutput:(AVCaptureOutput*)captureOutput
        didOutputSampleBuffer:(CMSampleBufferRef)sampleBuffer
        fromConnection:(AVCaptureConnection*)connection;
```

このメソッドには，引き数が 3 つある．1 つめは，アウトプットのインスタンス． 2 つめは，CMSampleBufferRef という型のオブジェクトで，ここからキャプチャされたデータを取り出す．3 つめは，AVCaptureConnection というクラスのオブジェクトで，このセッションにおけるインプットとアウトプットの接続状態を表すものだ．

先に，AVCaptureConnection を調べてみよう．このクラスを使うと，キャプチャされたビデオやオーディオの状態を知る事ができる．今回はビデオのキャプチャのため，このクラスが持っているプロパティのうち，以下のものに興味がある。

- videoMirrored
- supportsVideoMirroring
- videoOrientation
- supportsVideoOrientation

videoMirrored と supportsVideoMirroring は, ビデオが鏡像表示されているかどうかを表すもの．videoOrientation と supportsVideoOrientation は，ビデオデータの向きを表すものだ. これらの値によって, 取得されたデータをどう取り扱えばいいのかが分かる．iPhone4 のカメラでこれらの値を確認してた．すると，videoMirrored と supportsVideoMirroring は，常に NO．videoOrientation は，バックカメラだと AVCaptureVideoOrientationLandscapeLeft，フロントカメラだと AVCaptureVideoOrientationLandscapeRight という結果だった．つまり，ビデオデータは常に横向きで送られてくることになる．これはデバイスを縦に持っていても，データとしては強制的に横向きにされることを表している．画像処理を行うために UIImage などのインスタンスを作るときは，注意が必要だ[14].

3.5 CMSampleBuffer からの画像の取得

2 つめの引き数である CMSampleBufferRef を調べてみよう．これは，CM で名前が始まっていることから分かるように，Core Media フレームワークのオブジェクトになる．Core Media は低レベルのメディアデータに直接アクセスするためのフレームワークだ．CMSampleBufferRef は，そのメディアデータのバッファを表すものになる．

このバッファからは，ビデオデータの場合 CVImageBuffer オブジェクトを取り出す事ができる．似たような名前が続いて混乱するかもしれないが，今度は CV で名前が始まっているので，Core Video フレームワークのオブジェクトになる．CVImageBuffer からは，ビデオ画像のピクセルデータを取得する事ができる．

ピクセルデータさえ取得できれば，これを Core Graphics の CGImage にすることができ，さらに UIImage にすることができる．こうなれば，自由に画面に貼付けて使えるだろう[15].

3.6 ビデオカメラの実現

AVCaptureVideoDataOutput のデリゲートメソッドを，次のように実装する．

● ソースコード

ViewController.m

```
- (void)captureOutput:(AVCaptureOutput*)captureOutput
            didOutputSampleBuffer:(CMSampleBufferRef)sampleBuffer
            fromConnection:(AVCaptureConnection*)connection
{
    // イメージバッファの取得
①   CVImageBufferRef      buffer;
    buffer = CMSampleBufferGetImageBuffer(sampleBuffer);

    // イメージバッファのロック
②   CVPixelBufferLockBaseAddress(buffer, 0);

    // イメージバッファ情報の取得
    uint8_t*       base;
    size_t         width, height, bytesPerRow;
    base = CVPixelBufferGetBaseAddress(buffer);
    width = CVPixelBufferGetWidth(buffer);
    height = CVPixelBufferGetHeight(buffer);
    bytesPerRow = CVPixelBufferGetBytesPerRow(buffer);

    // ビットマップコンテキストの作成
    CGColorSpaceRef colorSpace;
    CGContextRef      cgContext;
    colorSpace = CGColorSpaceCreateDeviceRGB();
④   cgContext = CGBitmapContextCreate(
                base, width, height, 8, bytesPerRow, colorSpace,
                kCGBitmapByteOrder32Little | kCGImageAlphaPremultipliedFirst);
    CGColorSpaceRelease(colorSpace);
```

続き

```
    // 画像の作成
    CGImageRef    cgImage;
    UIImage*      image;
    cgImage = CGBitmapContextCreateImage(cgContext);
    image = [UIImage imageWithCGImage:cgImage scale:1.0f
                orientation:UIImageOrientationRight];
    CGImageRelease(cgImage);
    CGContextRelease(cgContext);

⑤   // イメージバッファのアンロック
    CVPixelBufferUnlockBaseAddress(buffer, 0);

    // 画像の表示
    _imageView.image = image;
}
```

① CMSampleBuffer から CVImageBuffer を取得しよう．これには，CMSampleBufferGetImageBuffer という関数を利用する．

② CVImageBuffer を取得したら，処理を始める前にロックしなければならない．ロックをしないと，カメラから送られてくるデータで次々と書き換えられてしまう事になる．そこで，CVPixelBufferLockBaseAddress という関数を使ってロックする．

③ CVImageBuffer の各種情報を取得する．取得しているので，データのベースアドレス，横幅ピクセル数，縦幅ピクセル数，1 行あたりのバイト数，だ．

④ これらの情報がそろえば，Core Graphics のイメージコンテキストを作成する事ができる．CGBitmapContextCreate を使って，CGContext を作成しよう．
後は，CGContext から CGImage を取得し，そこから UIImage を作成する．ただし，このとき 1 つ気をつける事がある．それは，ビデオデータの向きだ．先ほど説明したように，ビデオデータの向きは AVCaptureConnection の videoOrientation から取得できるが，それに応じた変換を行わなくてはいけない．そのためだろうか，iOS4.0 から UIImage の初期化メゾットが追加されている．imageWithCGImage:scale:orientation:というものだ．引き数に画像データの向きを指定して，それに合わせて変換してくれるものだ．ここでは，iPhone 4 のバックカメラを想定して，UIImageOrientationRight を指定している．

⑤ UIImage の取得ができたら、CVImageBuffer を忘れずにアンロックしておこう．これで次のデータが送られてくる．

取得した UIImage を画面上に貼付ければ，ビデオ画像が表示される事になる[16].

3.7 リアルタイム加工カメラアプリ作成

カメラの各ドットの RGB 情報の数値を弄ることでリアルタイム加工をしている．ここに，「AudioToolbox.framework」を追加する．追加の仕方は，先ほどと同じである．追加したら，ViewController.h を開き，それぞれ以下のように追記する[17].

- ソースコード

ViewController.h

```
#import <UIKit/UIKit.h>
#import <AVFoundation/AVFoundation.h>

@interface VideoCameraViewController : UIViewController
① <AVCaptureVideoDataOutputSampleBufferDelegate>
{
    AVCaptureSession*    _session;
②   IBOutlet UIImageView*    _imageView;
③   NSInteger    _frames;
④   float*    _total;

}

@end
```

① プロトコルを実装を宣言する．
② ライブラリペインから，エディタエリアに Image View を貼り付け，Outlet 接続する．
③ 64 ビットアプリにおいて，確実に 64 ビット幅の整数値を必要とするため，NSInteger というクラスを使う．そのあとに，起動してからフレームをだす．
④ 指定された要素を左または右に寄せて配置する際に float を用いる．そして，カメラから入出力される RGB 値の累積を記録する（これをフレーム数で割れば平均値が求まる）．

ViewController.m

```
        // セッションの作成
        _session = [[AVCaptureSession alloc] init];
        [_session addInput:deviceInput];
        [_session addOutput:dataOutput];

        // セッションの開始
        [_session startRunning];
①   _frames = -10;
    }
```

① 最初の数フレームは無視する.

ViewController.m

```
// ビットマップコンテキストの作成
CGColorSpaceRef colorSpace;
CGContextRef    cgContext;
colorSpace = CGColorSpaceCreateDeviceRGB();
cgContext = CGBitmapContextCreate(
        base, width, height, 8, bytesPerRow, colorSpace,
        kCGBitmapByteOrder32Little | kCGImageAlphaPremultipliedFirst);
CGColorSpaceRelease(colorSpace);

if (_frames == 1) {
    _total = (float*)malloc(sizeof(float) * height * width * 3);
}
① for (int j = 0; j < height; j++) {
②     for (int i = 0; i < width; i++) {
③         UInt8 *tmp3 = base + j * bytesPerRow + i * 4;
          UInt8 r = *(tmp3 + 2);
          UInt8 g = *(tmp3 + 1);
          UInt8 b = *(tmp3);

          unsigned long p = ((j * width) + i) * 3;
          if (_frames > 0) {
              if (_frames == 1) {
                  _total[p] = r;
                  _total[p + 1] = g;
                  _total[p + 2] = b;
              } else {
                  _total[p] += r;
                  _total[p + 1] += g;
                  _total[p + 2] += b;
              }
              *(tmp3 + 2) = _total[p]/_frames;
              *(tmp3 + 1) = _total[p + 1]/_frames;
              *(tmp3) = _total[p + 2]/_frames;
          }
    }
}
_frames++;
```

① 画面全てのドットの RGB データを取得，加工，置き換えを行う．
② カメラ画像の x,y（ここでは j,i だが）座標のポインタを取得している．
③ 座標の RGB のデータを取得する．取得した RGB データを加工し，またもとのポインタに書き換えてやることで画像を加工している．

参考文献

[12] 実践！iPhone アプリ開発．ビデオカメラの作り方(1)．セッションの作成．
http://news.mynavi.jp/column/iphone/040/, (参照 2015-11-14)
[14] 実践！iPhone アプリ開発．ビデオカメラの作り方(2)．ビデオ画像の表示．
http://news.mynavi.jp/column/iphone/041/, (参照 2015-11-14)
[17] リアルタイム加工カメラアプリの作り方 iOS.
http://nyaonyaokun.hatenablog.com/entry/2014/12/28/044344, (参照 2015-1-25)
[20] iOS のカメラ機能を使う方法まとめ．
http://dev.classmethod.jp/smartphone/ios-camera-intro/, (参照 2015-11-21)

第４章 マネジメント

4.1 本章の構成

本章では，本研究と関係のあるマネジメントの定義について述べる．

4.2 プロジェクト品質マネジメントとは

プロジェクトが取り組むべきニーズを満足するために，品質方針，品質目標，品質責任などを決定する母体組織のプロセスと活動を含む．プロジェクトの状況に合わせて組織の品質マネジメント・システムを実施するために方針と手順を使用し，母体組織の名の下に行われる継続的プロセス改善活動を支える．プロダクト要求事項を含むプロジェクト要求事項が確実に満たされ，かつ妥当性確認されるように働きかける．

プロジェクト品質・マネジメント・プロセスには以下の 3 つのプロセスがある．

(1) 品質マネジメント計画

インプット

- プロジェクトマネジメント計画書
- ステークホルダー登録簿
- リスク登録簿
- 要求事項文書
- 組織体の環境要因
- 組織のプロセス資産

ツールと技法

- 費用便益分析
- 品質コスト（COQ）
- QC 七つの道具
- ベンチマーキング
- 実験計画法
- 統計的サンプリング
- その他の品質計画ツール
- 会議

アウトプット

- 品質マネジメント計画書
- プロセス改善計画書
- 品質尺度
- 品質チェックリスト
- プロジェクトマネジメント計画書更新版

(2) 品質保証

インプット

・品質マネジメント計画書
・プロセス改善計画書
・品質尺度
・品質コントロール測定結果
・プロジェクト文書

ツールと技法

・品質マネジメントとコントロールのツール
・品質監査
・プロセス分析

アウトプット

・更新要求
・プロジェクトマネジメント計画書更新版
・プロジェクト文書更新版
・組織のプロセス資産更新版

(3) 品質コントロール

インプット

・プロジェクトマネジメント計画書
・品質尺度
・品質チェックリスト
・作業パフォーマンス・データ
・承認済み変更要求
・成果物
・プロジェクト文書
・組織のプロセス資産

ツールと技法

・QC 七つの道具
・統計的サンプリング
・検査
・承認済み変更要求のレビュー

アウトプット

・品質コントロール測定結果
・妥当性確認済み変更
・検証済み成果物
・作業パフォーマンス情報
・変更要求
・プロジェクトマネジメント計画書更新版
・プロジェクトマネジメント文書更新版
・組織のプロセス資産更新版

上記のプロセスは相互に相互に働きかけ，他の知識エリアのプロセスとの相互作用もある．プロジェクト品質マネジメントでは，プロジェクトのマネジメントと成果物のマネジメントの両方を取り扱う.成果物の性質にかかわらず,全てのプロジェクトに適用される．成果物の品質尺度や技法は，プロジェクトで生み出される成果物の個々のタイプに固有なものである．

プロジェクト品質マネジメントの基本的な取り組みは，国際基準化機構(ISO)の品質基準と互換性をもたせようとしている．すべてのプロジェクトは品質マネジメント計画書をもつべきである．プロジェクト・チームは品質マネジメント計画書に従うと共に，同計画書を順守していることを示すデータを保持する必要がある．ISO との親和性をもたせるため，最新の品質マネジメント手法では，多様性を最小限に抑え，定義された要求事項を満たす結果を実現するように努めている．以下に示す点が重要であることを認識してる．

- 顧客満足
- 検査よりも予防
- 継続的改善 PDCA(plan-do-check-act)
- 経営者の責任
- 品質コスト(COQ)

4.3 プロセス概要

(1) 品質マネジメント計画

プロジェクトおよびその成果物の品質要求事項または品質標準，あるいはその両方を定め，プロジェクトで品質要求事項または品質基準，あるいはその両方を順守するための方法を文書化するプロセスである．品質の計画は，ほかのプロセスと並行して遂行すべきである．たとえば，成果物が所定の品質基準を満たすように提案された変更により，コストやスケジュールの調整，およに他の計画への影響についての詳細なリスク分析を行う必要がある．品質マネジメント計画書を作成するためにはプロジェクトマネジメント計画書を用いる．作成に用いる情報には以下の項目が含まれるが，これらに限定されるものではない．

- スコープ・ベースライン
- スケジュール・ライン
- コスト・ベースライン
- その他のマネジメント計画書

(2) 品質保証

適切な品質基準と運用基準の適切を確実に行うために，品質の要求事項と品質コントロールの測定結果を監査するプロセスである．主な利点は，品質プロセスの改善を促進することにある．品質保証は，計画プロセスを通して欠陥を予防したり，作業実施中に欠陥を検出したりすることにより，品質が確かなものであるという状態を作り出すことに貢献する．品質プロセスでは，プロジェクトの品質マネジメント計画書に定義された計画済みの体系的な行動とプロセスを実施する．品質保証を展開することで，将来のアウトプットまたは未完成のアウトプット，更新中の作業が，特定された要求事項や期待を満たすという確信をもつようになる．また，品質マネジメント計画プロセスと品質コントロール・プロセスのツールと技法を使用する．次に示す，その他のツールを使用することもできる．

- 親和図
- PDPC 法(Process Decision Program Charts)
- 連関図
- ツリー・ダイアグラム
- 優先順位マトリックス
- アクティビティ・ネットワーク図
- マトリックス・ダイアグラム

(3) 品質コントロール

パフォーマンスを査定し．必要な変更を提案するために品質活動の実行結果を監視し，記録するプロセスである．主な利点は次の通りである．

I. 粗悪なプロセスやプロダクト品質の原因を特定し，それを排除するための処置を提言あるいは実行する．

II. プロジェクトの成果物および作業が，最終的な受け入れのために主要なステークホルダーが特定した要求事項を満たしていることの妥当性確認を行う

品質コントロール・プロセスでは複数の技法とタスクを用いて，作成されたアウトプットが要求事項を満たすかを検証する．品質保証は，プロジェクトの計画フェーズと実行フェーズでステークホルダーの要求事項が満たされる確信を得るために使用され，品質コントロールはプロジェクトの実行フェーズと終結フェーズで信頼性の高いデータに基づいてスポンサーや顧客の受入基準が満たされたことを公式に明らかにするために使用される．

プロジェクトマネジメント・チームは，品質コントロールのアウトプットに含まれるデータを評価するうえで必要な，統計的コントロール・プロセスの実務的知識を持ち合わせている．さまざまな事項のうち，以下の対になっている用語の違いを知ることは，役に立つ．

- 予防（プロセスからエラーをなくす）と検査（顧客の手に渡るエラーをなくす）
- 計数サンプリング（結果の合否）と計量サンプリング（適合の度合いを連続的尺度を用いて測定した結果の等級付け）
- 許容誤差（許容可能な結果の指定範囲）と管理限界（統計的に安定したプロセスまたはプロセス・パフォーマンスで一般的に見られる変動の範囲）

第 5 章　結論

5.1 研究の結果

開発を行う前の，証明書署名要求(CSR)や iOS Developer Program から証明書ファイルを作成し，Mac に開発用証明書(CER)や App ID の登録を行い，環境開発に必要なツールを揃えることができた．本論文には記載していないが，ソフトウェア開発のための統合開発環境(IDE：Integrated Development Environment)である Xcode の用語や使い方を学ぶことができた．リアルタイムに動画を処理するスマートフォンアプリケーションは，動いているものを消すということが目的だが，その前の段階として，この開発に必要とも言える，カメラアプリケーションとビデオ動画アプリケーションを完成させ，実装に成功することができた. また, この開発における作り方やコードについて, 様々な知識を得ることができた．今回の研究目的である，リアルタイムに動画を処理するスマートフォンアプリケーションの開発については，カメラの各ドットの RGB 値の累積を記録し，フレーム数で割ることにより平均値が求められ，動いている物を消すことができた．しかし，RGB のコード変換やピクセル分解などは，まだまだ勉強不足だと実感し，わからないことが多いと感じた．

5.2 今後の課題

今回は，何の機能も付けず，端末をかざすだけで動いている物をリアルタイムで消すとういことだった．このアプリケーションの最終地点としては，ビデオ録画アプリケーションに今回開発した機能を加える，リアルタイムに処理できるうえに，カメラロールに保存でき，データが残るということが必要であると考える．

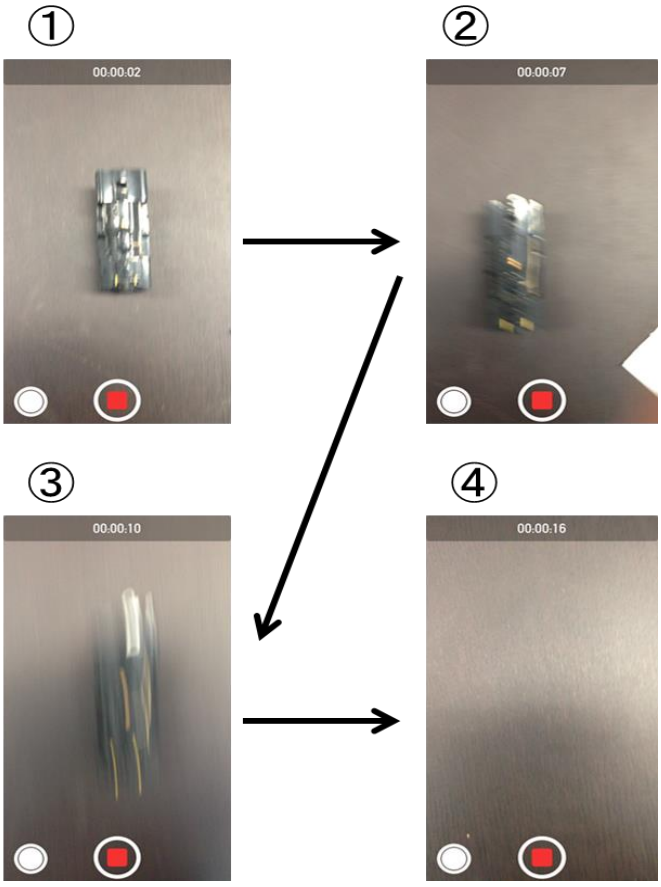

図 5-1 イメージ図

5.3 考察

実際に RGB の平均値を取ることで, 動くものを消すことができた. 長く時間を取れば, 最頻値の方が正確に処理できると思うが, コマ数を記憶する為のメモリが多く必要になるが, リアルタイムに処理する際には, 平均値で十分であることが本研究で実証された.

5.4 謝辞

本論文作成にあたり, テーマの決定, 論文の書き方, 研究内容などすべてにおいて, 長期にわたって厳しくも熱意のあるご指導, ご鞭撻していただいた, 矢吹太朗准教授に厚く御礼申し上げます. 特に概要の書き方においても拙い私の論文を, 何度も読んでいただき, 指導していただいた矢吹太朗准教授に大変ご苦労をかけてしまいましたことにも心よりお詫びを申し上げたいです. その他, 助けていただいた多くの皆様心から感謝しております. ありがとうございました.