iOS授業3日目資料

はじめに

はじめに：今日やること

- ◆1.ARKitについて
- ◆2.色々な物体を表示してみよう
- ◆3.3Dモデルを使ってARで遊ぼう
- ◆4.画像トラッキングに挑戦してみよう
- ◆5.Git を使ってみよう

はじめに：今日の目標

◆ＡＲＫｉｔで卒業制作する人・何か作りたい人

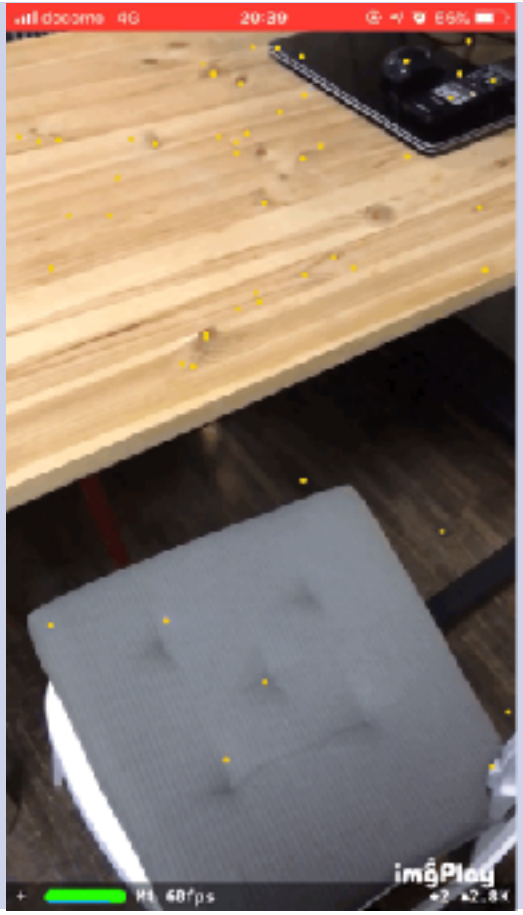
◆ＡＲＫｉｔの基礎や、３Ｄの基礎が身に付けばＯＫ！

◆Ｆｕｎｃｔｉｏｎ の使い方を身につけましょう！

◆それ以外の人

◆楽しみましょう！！！
◆わからなくてＯＫ！
◆Ｆｕｎｃｔｉｏｎ の使い方だけ覚えてください🤦

今日のアウトプット

物体の生成/加工	3Dモデルの使用	画像トラッキング
		

ARKitについて

ARKitの歴史について知る

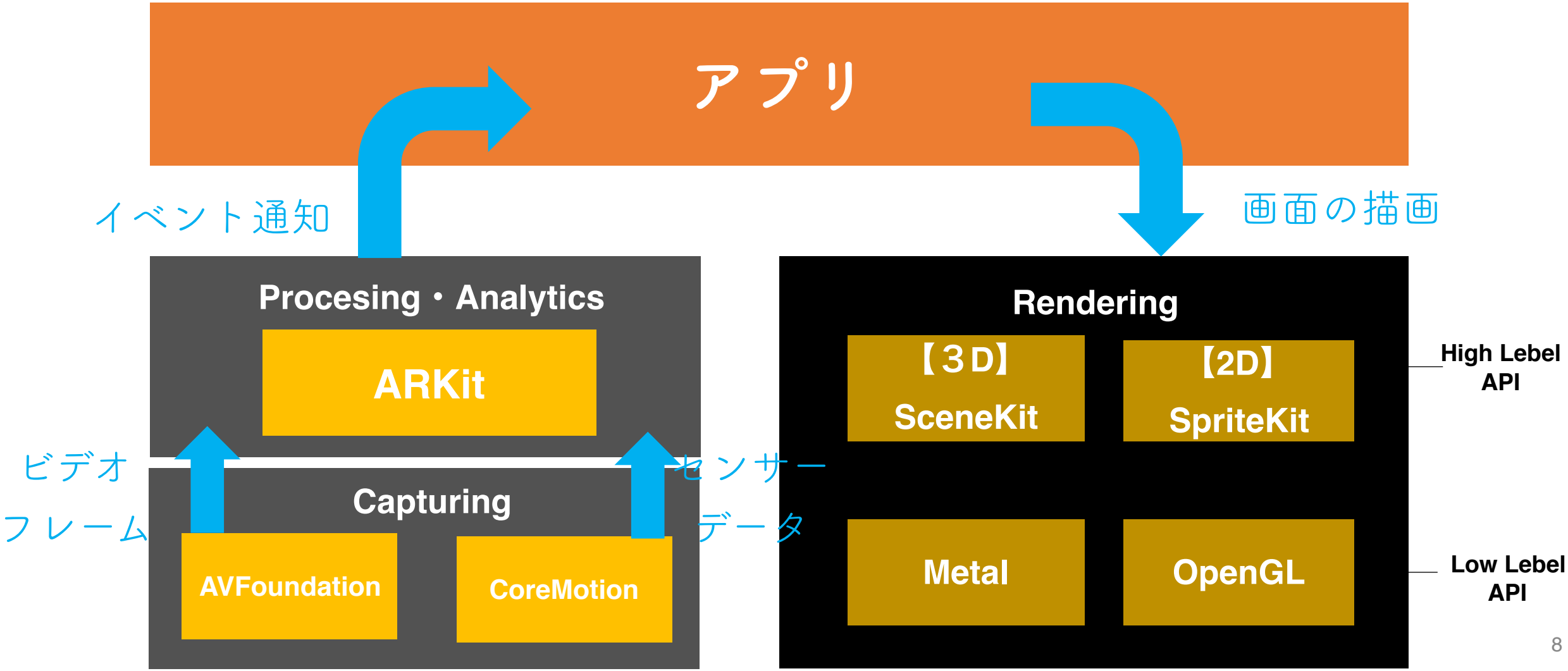
ARKitはiPhone・iPad向けのAR対応アプリのためのフレームワークです

ARKit1.0(iOS11～) 2017/6	ARKit1.5(iOS11.3～) 2018/3
水平平面の認識 3次元の点座標情報検出 自分の位置や向きを検出 現実世界のスケール検出 etc	壁など垂直平面の認識 画像認識 不規則形状の平面認識 解像度UP etc

ARKit2.0(iOS12～)2018/9	ARKit3.0(iOS13.0～) Now
3Dオブジェクトの認識 画像トラッキング 空間の共有・永続化 顔の認識　etc	人物オクルージョン モーションキャプチャ 複数人の顔認識 前面/背面カメラ同時使用 etc ※メジャーな新機能は A12 Bionicチップ搭載機のみ

iOSでのARのざっくりしたプロセス

ARKit自身はコア処理を担当し、レンダリング関連は他のFrameworkが担当



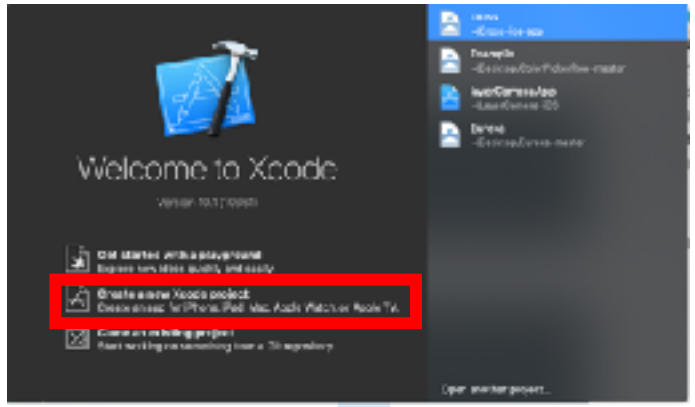
物体の生成・表示

【一緒にやってみよう】

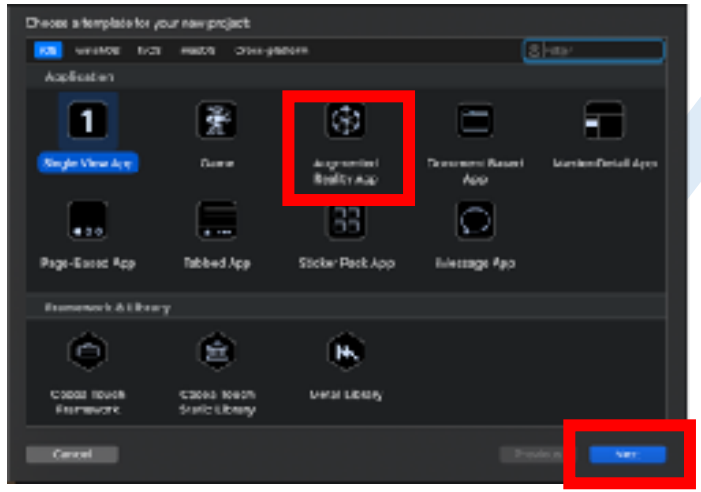
文字や図形を表示してみよう

ARのプロジェクトを作成してみよう

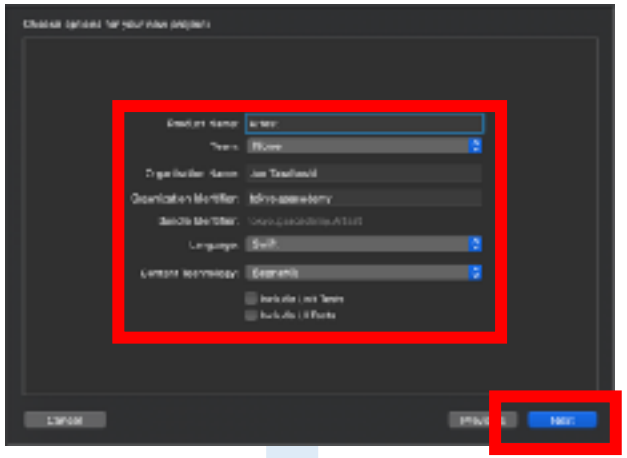
①Create a new Xcode Projectを選択



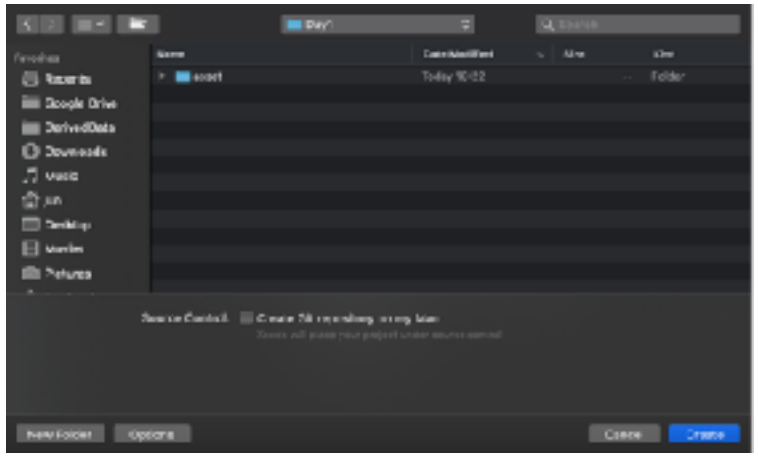
②Augmented Reality App＞Nextを選択



③プロジェクト情報を入力＞Nextを選択



④保存するフォルダを選択＞Createを選択



ARプロジェクトの初期ファイルを確認してみよう

項目名	項目の説明
AppDelegate.swift	アプリ全体のイベント（起動・終了等）を管理・制御するファイル
ViewController.swift	アプリの画面（View）を管理・制御をするファイル
Main.storyboard	アプリの画面のレイアウトや遷移などを管理するファイル
Assets.xcassets	画像などのアセットを管理するためのファイル（フォルダ）
art.scnassets	AR用のデータのアセットを管理するためのファイル（フォルダ）
LaunchScreen.storyboad	起動したときに表示される内容（スプラッシュ）のレイアウトなどを管理するファイル
Info.plist	アプリの共通的な設定を管理するファイル

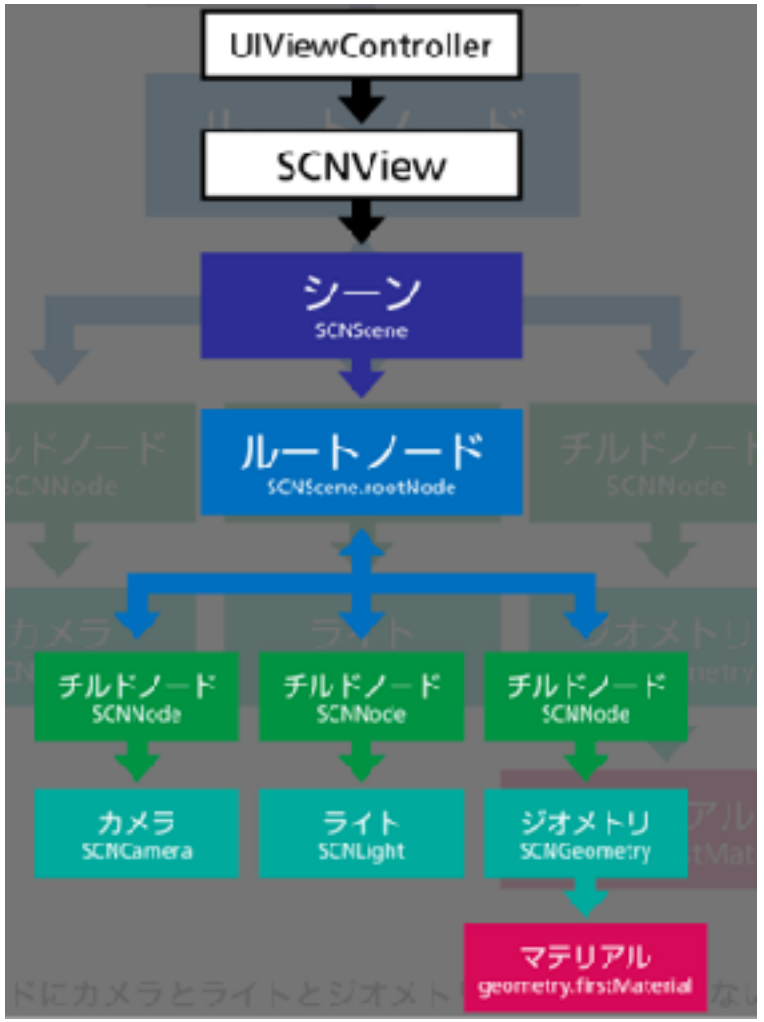
ARKitにおける3D空間を知る

【ポイント】

・ノードは形状や座標などの情報を持つ
ビルディングブロック

・空間は右のようなツリー構造になっており
シーンに必ず1つだけある
起点の「ルートノード」に
各オブジェクトが接続されているノードを、
「チャイルドノード」として追加していく

・空間にある全てのオブジェクトは
ノードに接続して管理されることになる



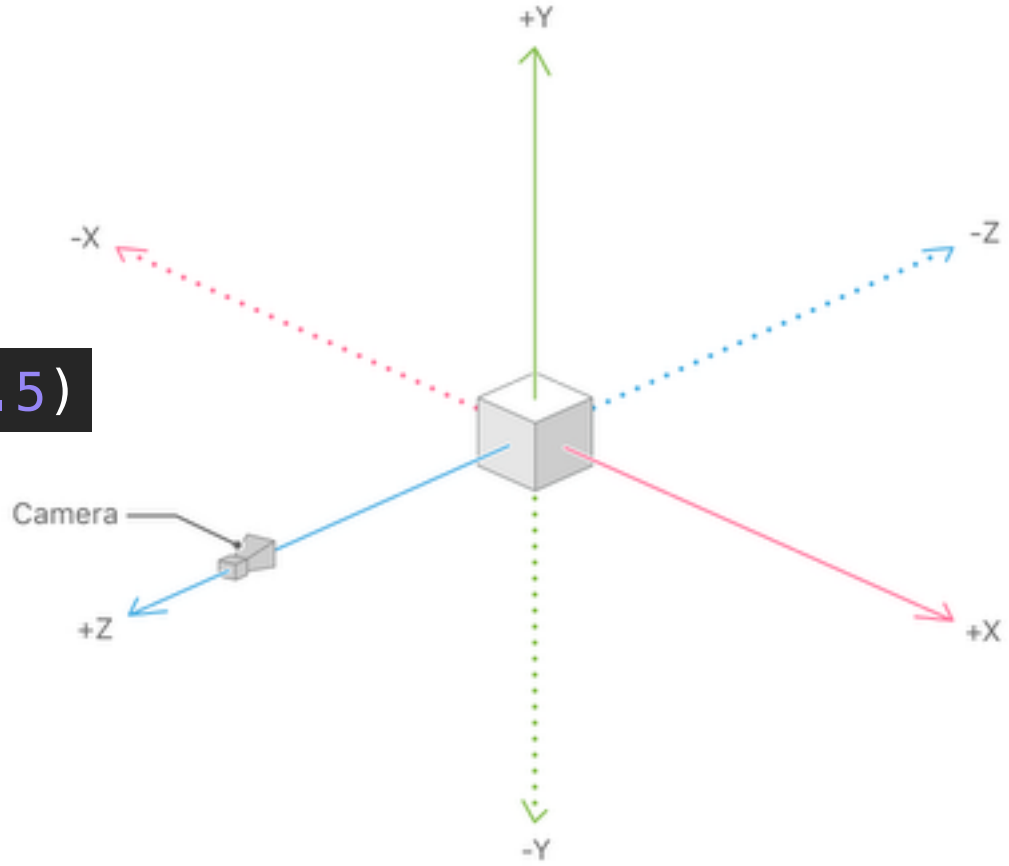
http://appleengine.hatenablog.com/entry/2017/05/30/192435

ARKitの座標について知る

ARKitでは主にSCNVector3で
座標を指定する

```
boxNode.position = SCNVector3(0, 0, -0.5)
```

Zのマイナスが奥なので
これを間違えがち



https://developer.apple.com/

デフォルトで
1が実世界の1mに相当

図形や文字を空間に表示する基本的な手順を知る

① シーンの作成

・全体のシーン[空間]を作る

② ジオメトリの作成・設定

・ジオメトリ[3Dオブジェクトの形状]を指定

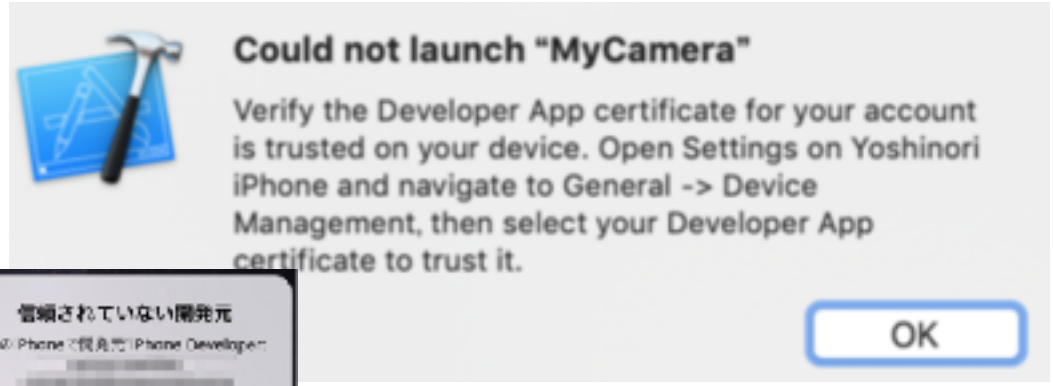
※マテリアル[外観]を設定する場合はここで

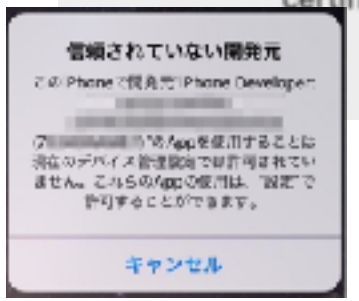
③ ノードの作成・設定

・②を格納するノードを作成

※ノードの位置や大きさを変えたい場合はここで

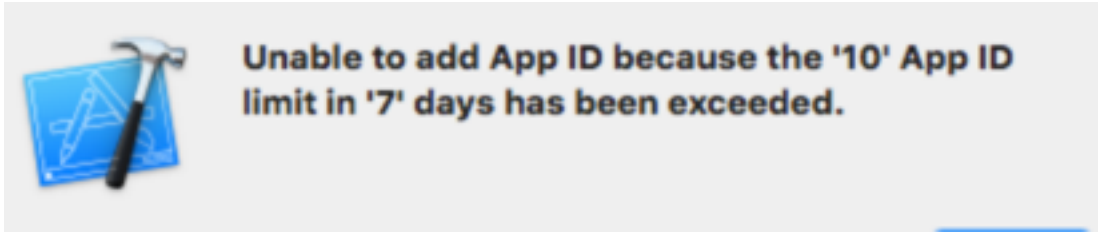
参考：実機ビルドできないとき





←こんなのがでたら
設定＞一般＞プロファイルとデバイス管理
で自分のAppleIDをタップしてあげる

あと無料アカウントだと
実機ビルド上限に達する場合があります→
新しいAppleIDを作る
すでに実機にインストールされている
BundleIDを使い回すなどして対処してください



【一緒にやってみよう】

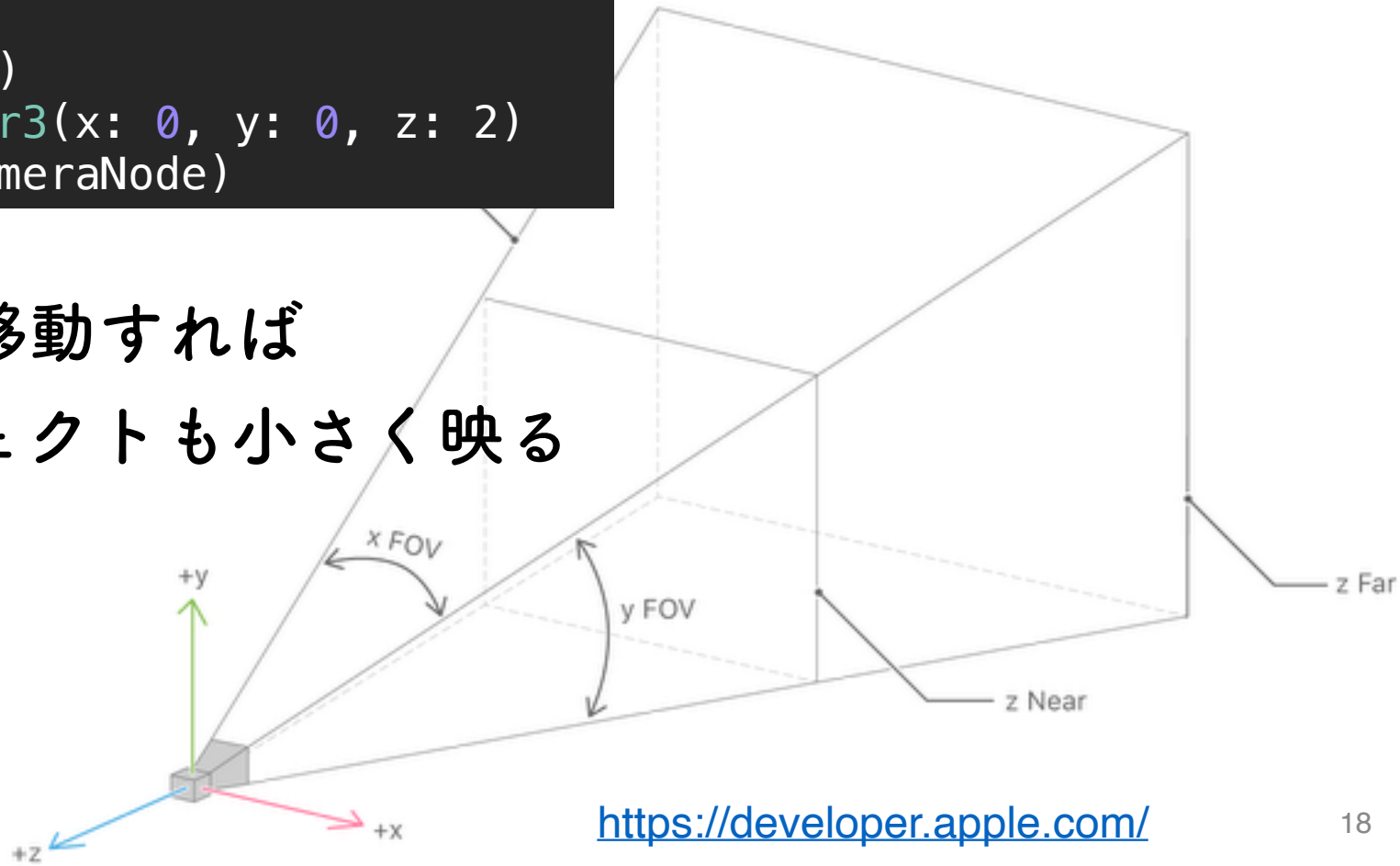
球体を地球っぽくしてみよう

参考：ARKitでのカメラについて

デフォルトのカメラの位置を変更することもできる

```
let cameraNode = SCNNode()
cameraNode.camera = SCNCamera()
cameraNode.position = SCNVector3(x: 0, y: 0, z: 2)
scene.rootNode.addChildNode(cameraNode)
```

※例えばカメラを手前に移動すれば
　眼の前にあったオブジェクトも小さく映る



https://developer.apple.com/

参考：ARKitでの照明について

自分で照明を作る場合は基本の２つ抑えておく

■①オムニライト[配置した位置から全ての方向を照らす光源]をあてる

```
let lightNode = SCNNode()
lightNode.light = SCNLight()
lightNode.light!.type = .omni
lightNode.position = SCNVector3(x: 0, y: 2, z: 2)
scene.rootNode.addChildNode(lightNode)
```

■②アンビエントライト[位置は関係なく単に画面全体を照らす環境光]

```
let ambientLightNode = SCNNode()
ambientLightNode.light = SCNLight()
ambientLightNode.light!.type = .ambient
ambientLightNode.light!.color = UIColor.darkGray
scene.rootNode.addChildNode(ambientLightNode)
```

※位置がないので
影は落ちない

(補足)デフォルトのライトを付ける場合はこれだけ

```
sceneView.autoenablesDefaultLighting = true
```

ここまでのおさらい

こんなことを学びました

①ARKitについて
②ARプロジェクトの作成方法
③SceneKit（空間の構成）について
④文字や図形を空間に表示する方法
⑤マテリアルの設定方法
⑥SCNActionでのアニメーション

３Ｄモデル(daeファイル)を扱ってみる

３Ｄモデルファイルについて

・iOSのSceneKitではscnファイルという形式のものを扱う

・scnファイル自体が一つの空間（シーン）になっている

・daeファイルはXcodeの機能によって
　scnファイルに変換することができる

・XcodeがUnityのようにオールマイティではないことに注意

＜参考＞よく使われる３Ｄデータの対応表

.obj	インポートできるがアニメーションをサポートしていない
.fbx	ライブラリを使えばインポートできる 公式ではない
.3ds/.max	ライブラリを使えばインポートできる 公式ではない

参考：Unityを使った開発について

実はUnity(+ ARKit Plugin) でもiOSアプリの開発が可能（実機ビルドはXcode）
主にゲーム開発などで使われています

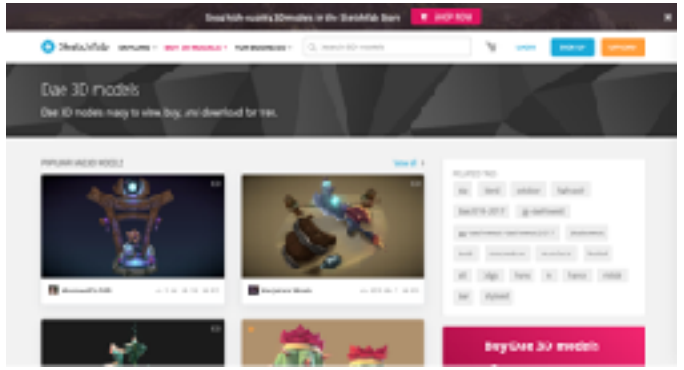
メリット	デメリット
・Unityの豊富なアセットを活用でき、 他の3Dツールとの相性もよい ・WEBや書籍などに3D表現のノウハウが ものすごく蓄積されている ・Unity自体はマルチプラットフォームなので 移植が楽になる	・Unity実行部分はC#で書く必要がある ・言語だけでなく Unity自体の操作や見方など学習コスト高め ・Unity側からiOSっぽいUIを出すのは手間がかかる ・ビルドが遅い/ワイヤレスデバッグできない ・Extension系などはやりづらい

目的とメリデメを考えて選択すること

参考：こんなところでdaeファイル拾えます

https://free3d.com/

https://www.turbosquid.com/



https://sketchfab.com/tags/dae

商用利用はライセンスをしっかり確認すること！

【一緒にやってみよう】

ピカチューをタップした平面においてみる

3Dモデルを空間に表示する基本的な手順を抑える

① シーンの作成

・全体のシーン[空間]を作る

② 3Dモデルのシーンを作成

・ファイルを指定してシーンを読み込み（作成）

③ ノードの作成・設定

・新しくノードを作成して、②についてるノードを付け替える
※ノードの位置や大きさを変えたい場合はここで

④ シーンの設定

・①のルートノードに③のノードを紐付ける

要件

① 平面を検出できるようにする

② タップを検知して何かできるようにする

③ ②の中で、平面のタップされた箇所を習得して
シーンにアンカーを追加する処理を作る

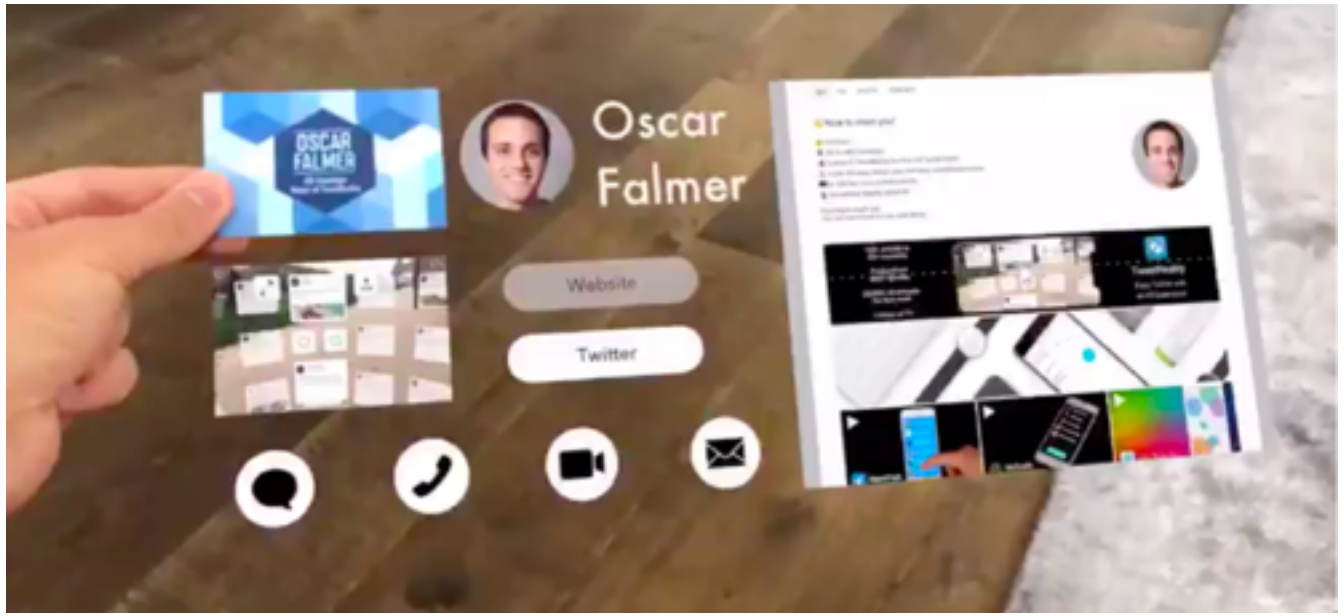
④ ③の追加を自動で検知してくれる関数を使って
アンカーの上にモデルを置く

※アンカー：実世界の位置と向きを持った目に見えないオブジェクト
…場所を固定するための「いかり」のようなもの

画像トラッキングをする

画像トラッキング機能について

ARKit2.0から現実空間の画像を検出・追跡し、
それをアンカーとして
オブジェクトを配置できることができるようになった
（1.5で画像検出自体はできた）



https://twitter.com/OsFalmer/status/1008736572185903105

参考：画像検出機能と何が違うのか？

画像検出は「ARWorldTrackingConfiguration」を使用し
トラッキングは「ARImageTrackingConfiguration」を使用する
これによって機能に差がでている

前者は現実空間を検出するコンフィギュレーションなので
現実の面を検出して、そこにある画像やオブジェクトを検出するが
後者は登録した既知の2D画像だけ検出して周りの面は検出しない

結果、パフォーマンスに差がでる
前者は静止画像しか認識できないが、
後者はトラッキングできる

【一緒にやってみよう】

墓からゾンビを呼び出そう

要件

① 認識させる画像を登録

② ①を認識してアンカーを自動で検知してくれる
　関数を使って、新しいノードを返す処理を用意

③ ②で検知したアンカーを使って
　色を塗った平面のノードを作る

④ ③で作ったノードにゾンビを生やす

参考：ARKitでの回転ついて

ARKitでは以下の２種類の回転方法がある

①SCNVector4での回転（x/y/zの軸と角度を設定）

```
boxNode.rotation = SCNVector4(1, 0, 0, 0.25 * Float.pi)
```

② eulerAngleでの回転

```
boxNode.eulerAngles.x = .pi / 4
```

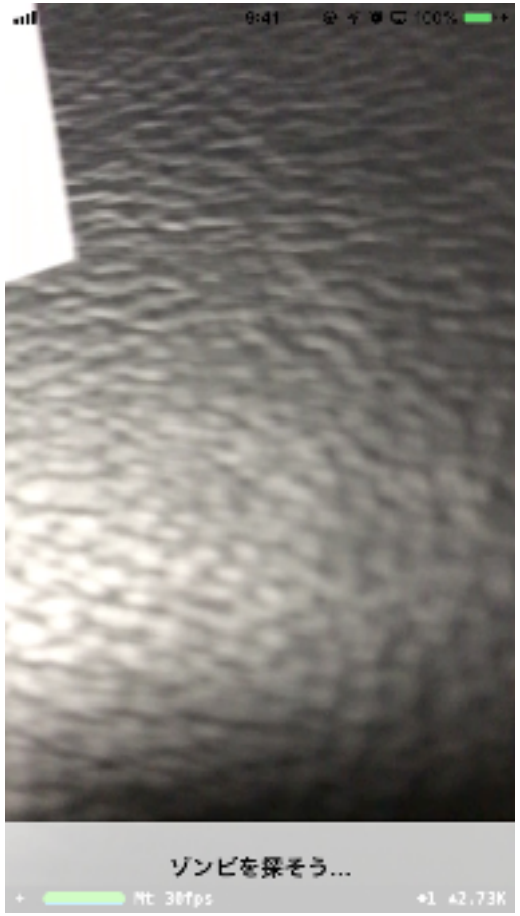
個別に設定

```
let d = Float.pi / 4
boxNode.eulerAngles = SCNVector3(d, 0, 0)
```

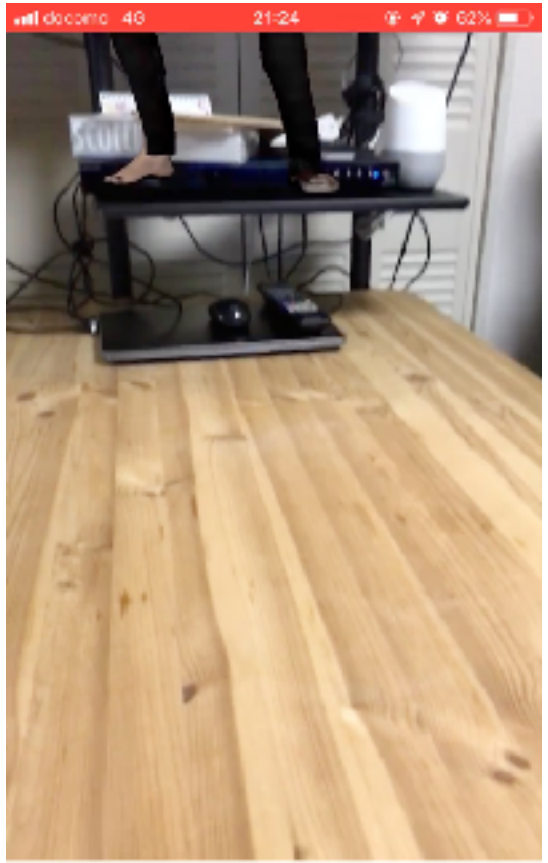
まとめて設定

参考：少し工夫するだけでこんなこともできる

ゾンビと話すアプリ



ゾンビを倒すアプリ



ARKitについて探求したい人

【まずは公式ドキュメント】

日本語化されてる部分も思いの外多いので

まずは全部しっかり読むこと、絶対にあとで役に立つ

https://developer.apple.com/jp/documentation/arkit/

【Udemy】

さくっと理解したい方はこちら

https://www.udemy.com/course/kboy-arkit/

その他公式やGitHubにあがってるコードを見ながら

面白そうなユースケースを探して、コードを見てみるのが一番勉強になるかも

【次回までの課題】

ARKitを使って面白いものを作ってくる

＜例＞

・地球のやつに太陽系を色々追加してみる/ タップで惑星の情報がでる…etc

・シューティングゲームを作ってスコアや時間・難易度の管理…etc

・免許証を認識して何か出してWEBへ飛ばすようなアプリ

・空間に落書きして保存できるアプリ

・機械学習モデルを使って現実空間のものに何かを表示する

【次回の予告】

ToDoアプリを作って
TableViewとSwiftともっと仲良くなる

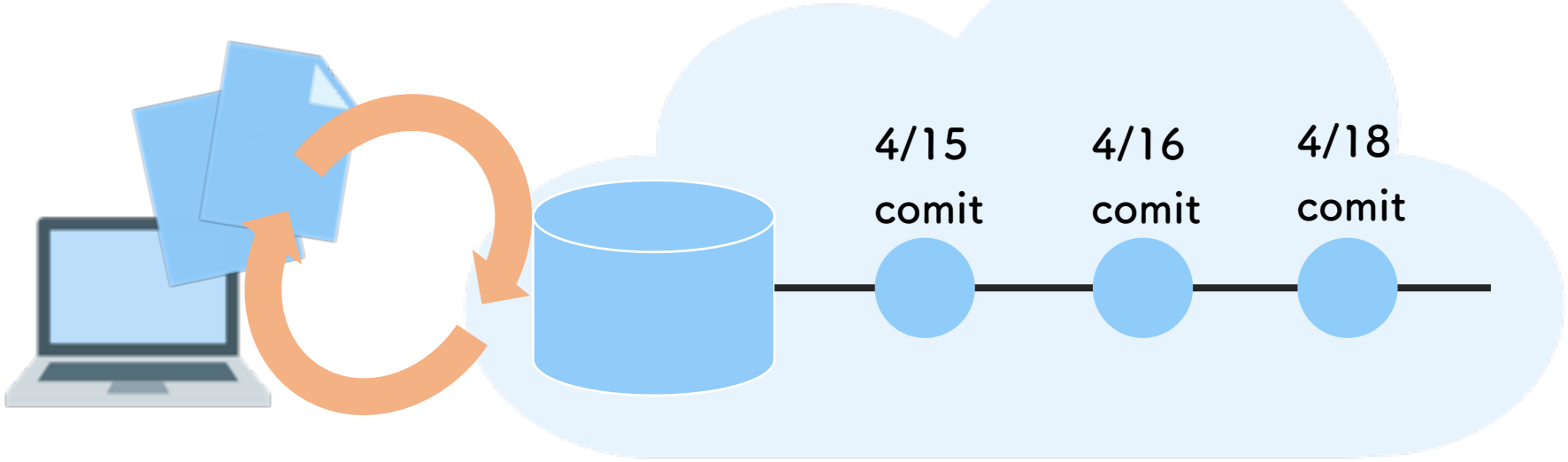
複数回の授業を通して
どんどんアプリを育てていきます

※今回は特に事前準備はいりません

Gitでのバージョン管理について学ぶ

Gitについて知る

ローカルのファイルなどに発生した変更を記録して
その履歴を管理するためのバージョン管理ツールのこと

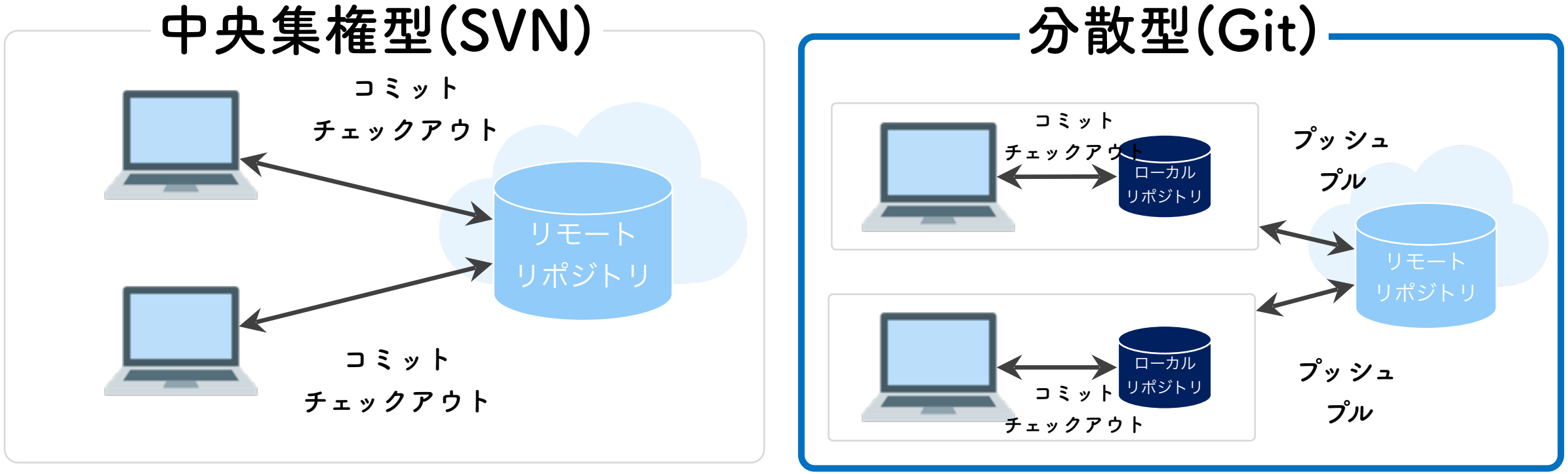


履歴を記録しておくことで、
昔の状態に戻したり過去の履歴を確認するのが簡単になる
というメリットがあります

Gitのバージョン管理の特徴

リモートリポジトリのみで中央集権的に管理するのではなく

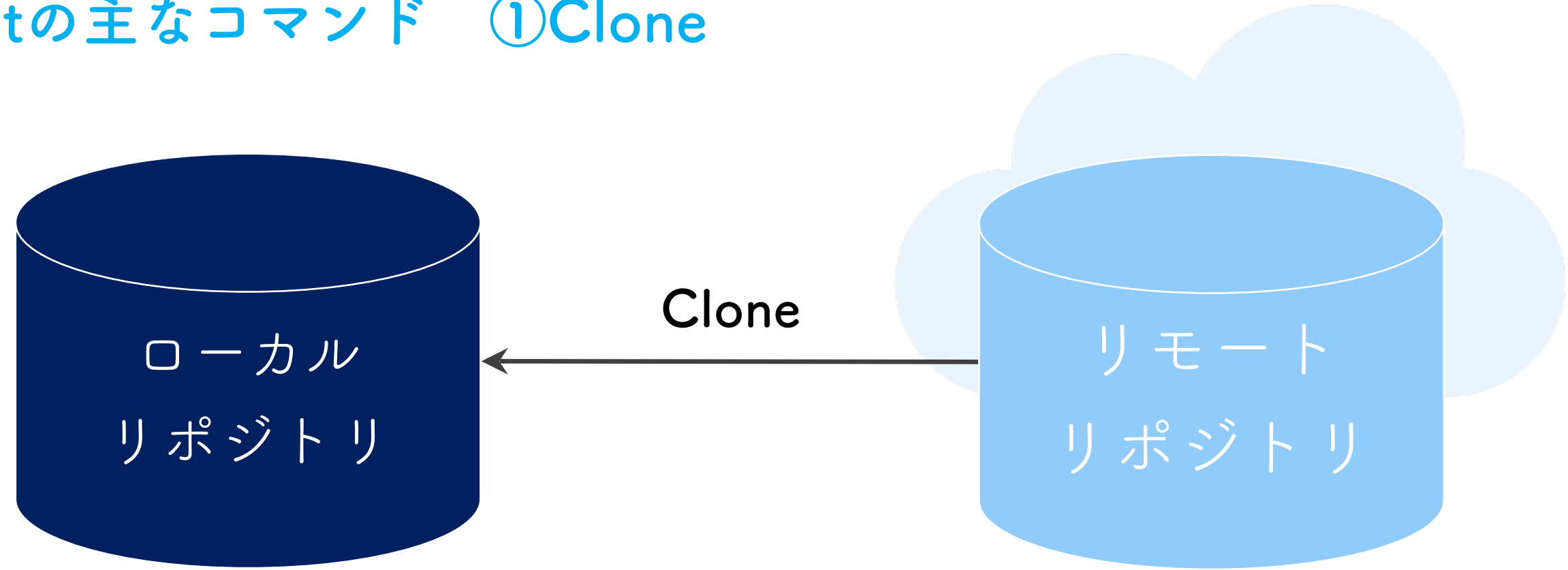
ローカルリポジトリというものを使う「分散型」というのが特徴



ローカルでソースコードの管理ができるため、

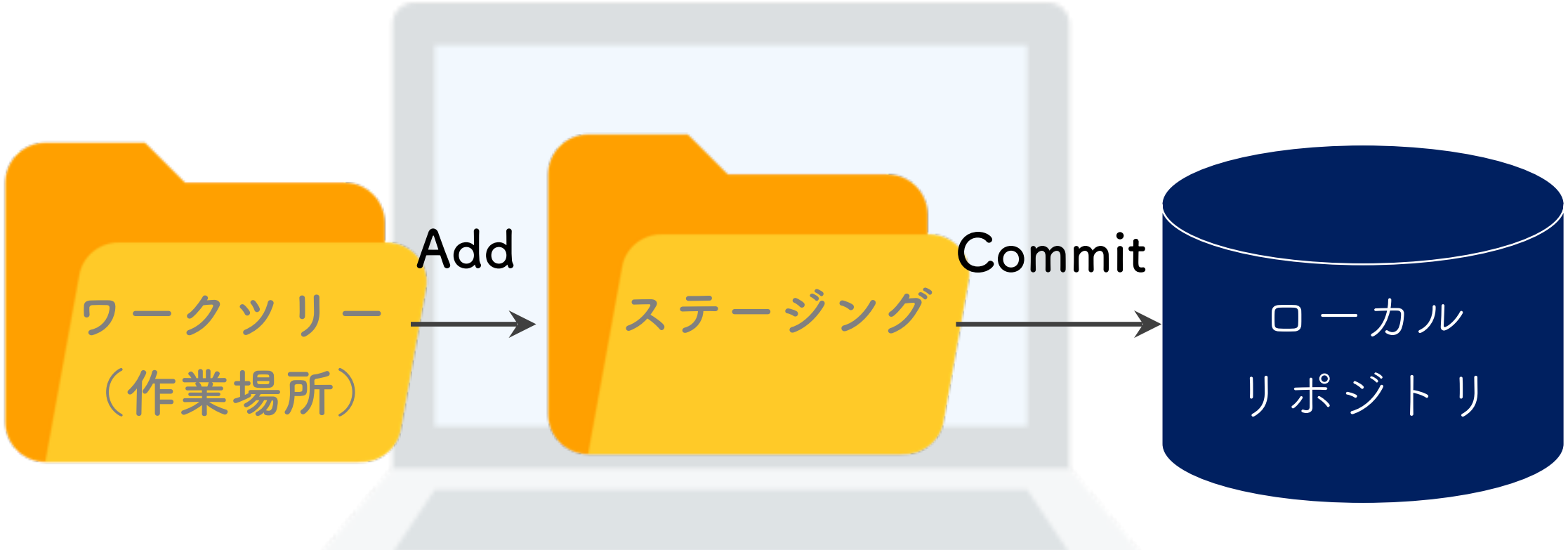
他の開発者に影響を与えず、自由に色々な作業ができます

Gitの主なコマンド　①Clone



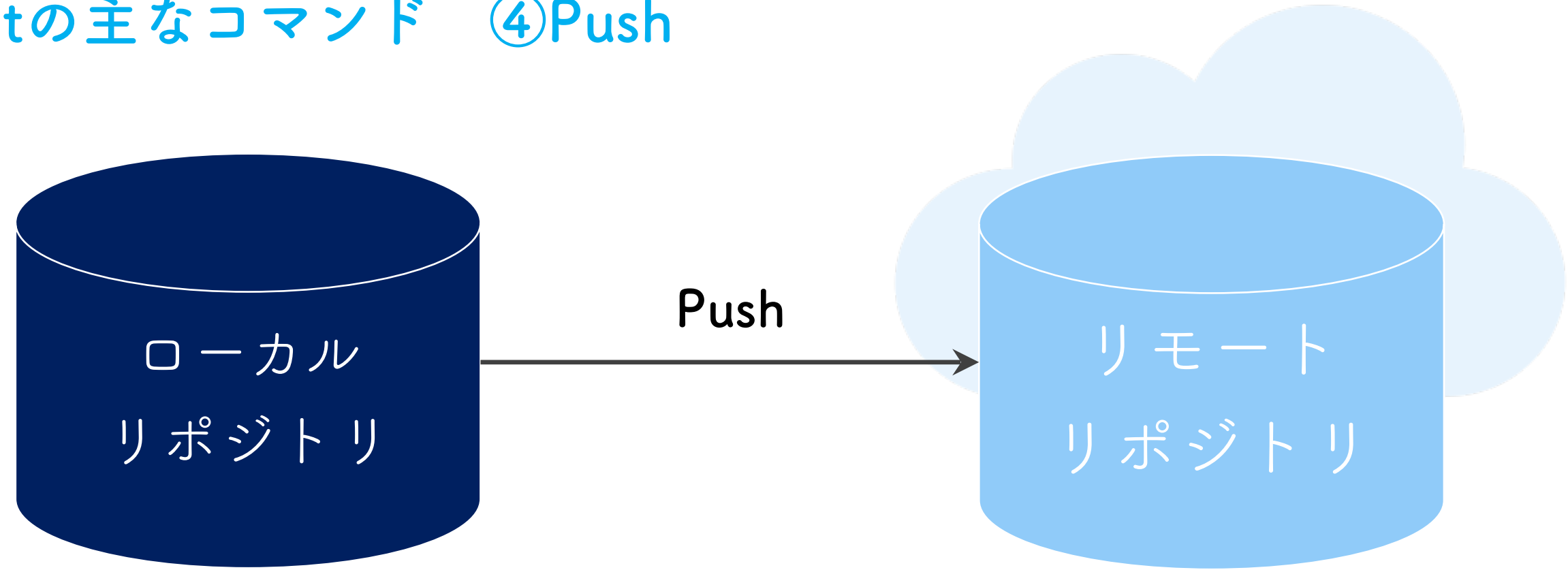
Cloneコマンドを使うと、リモートリポジトリを
手元のPCにコピーすることができます

Gitの主なコマンド　②Add ③Commit



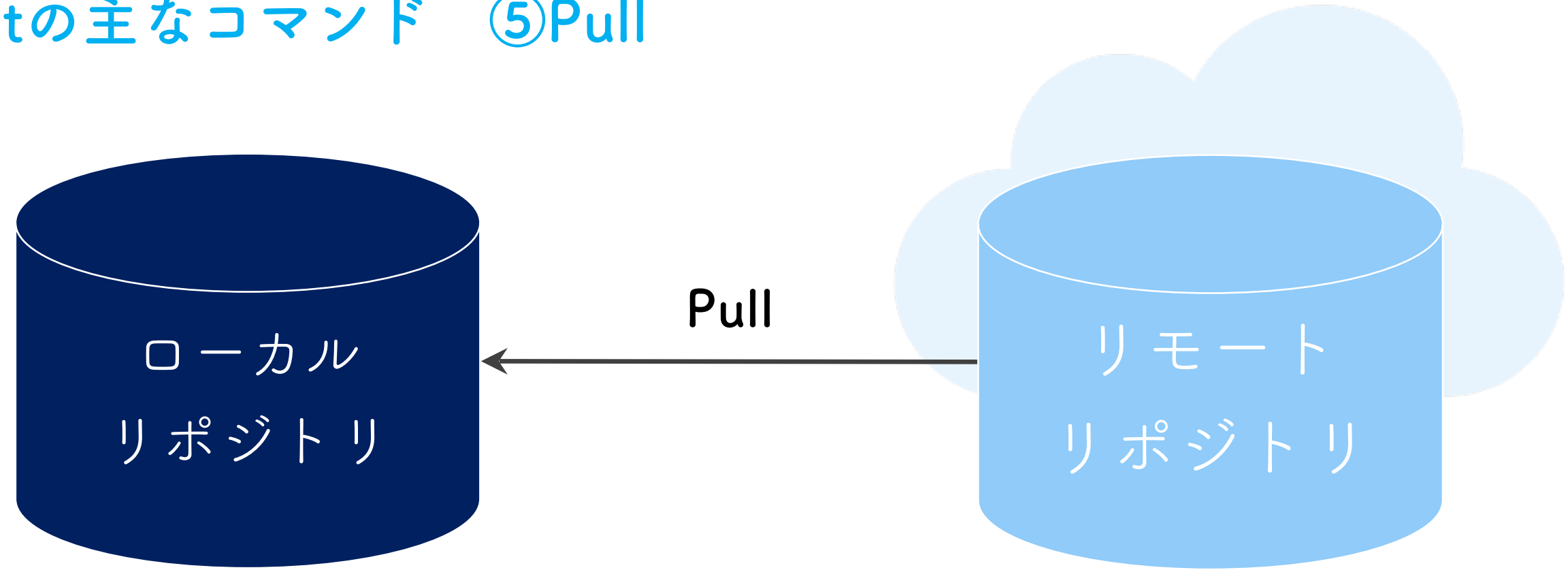
Addコマンドを使ってコミット対象を決めて、
Commitコマンドで、変更履歴をローカルリポジトリに保存します
「写真を撮る=Commit 」「撮影台に乗せる=Add」というイメージ

Gitの主なコマンド　④Push



Pushコマンドを使うと、リモートリポジトリに
ローカルの更新内容を同期させることができます

Gitの主なコマンド　⑤Pull

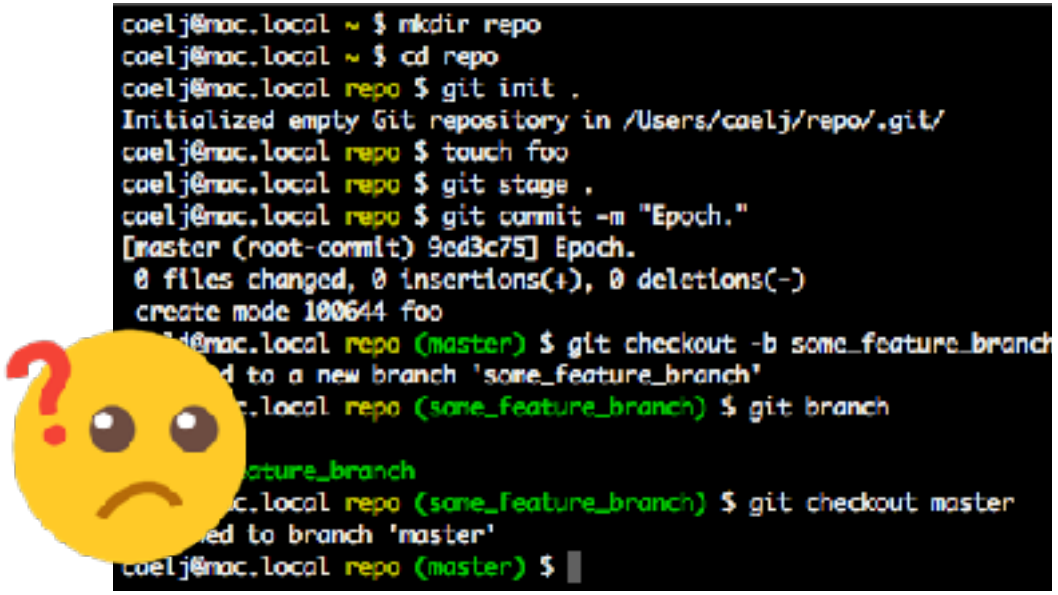


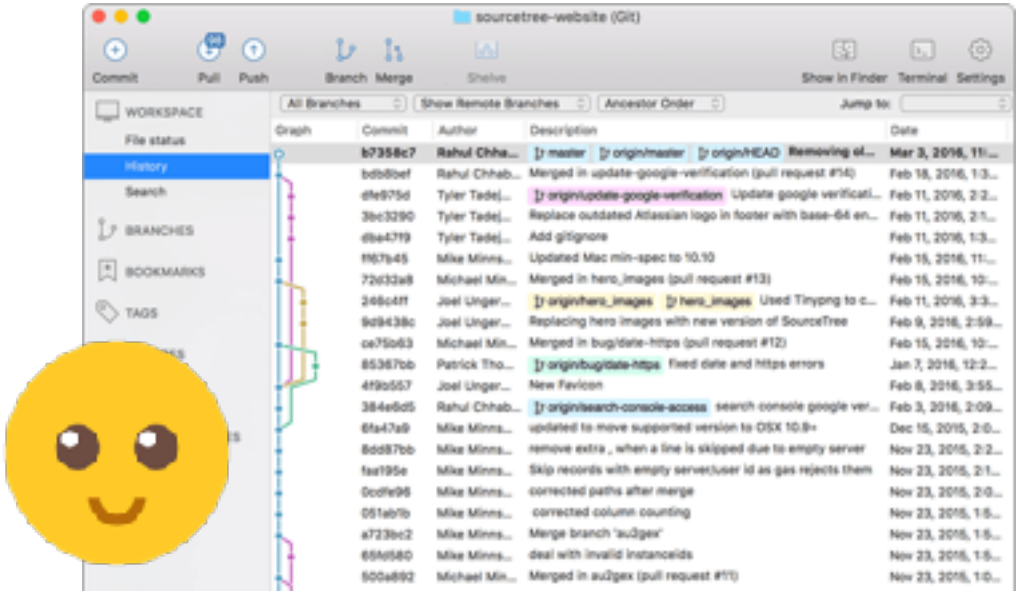
Pullコマンドを使うと、ローカルリポジトリに
リモートの更新内容を同期させることができます

Gitコマンドをどうやるか？

エンジニアがよく開いている黒い画面(ターミナル)で
コマンドを打つことができます

授業ではターミナルを使わず、
わかりやすいUIでボタンをポチポチするだけでコマンド実行できる
GUIのGitクライアント「SourceTree」を使います Sourcetree





補足：GUIでやるデメリットってあるの？

結論特にありません😂

CUIでやると

- 作業手順の共有が楽（コマンドシェアするだけ）
- エラー対処しやすい（情報がいっぱいある）
- 作業を自動化するのが楽
- かっこいい（8割）

今後エンジニアとしてやっていく予定の方は
CUIベースの作業も必ず習得しておきましょう

リモートリポジトリはどこで管理するのか？

Gitのホスティングサービス（リモートリポジトリを保存する先）は
GitHubをはじめとした様々なサービスがあります

授業では世界で最も有名な「GitHub」を使います
プライベートリポジトリも無料になりました😍

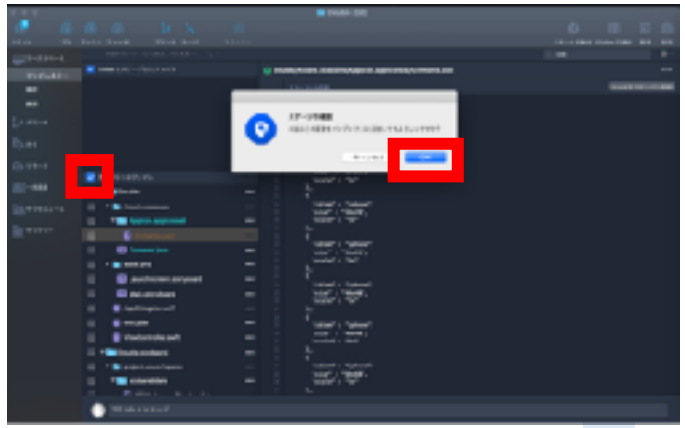
【一緒にやってみよう】

・新しくプロジェクトを作成する
・ローカル/リモートリポジトリを登録・作成する
・Add->Commit->Pushの操作を行ってリモートリポジトリに保存する

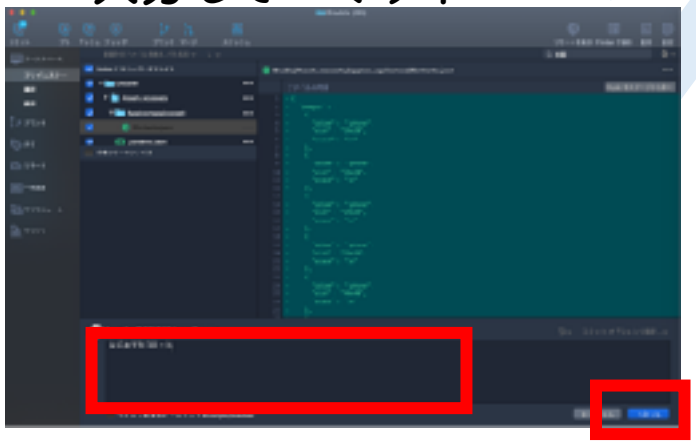
これから課題提出はGitHub使うので
頑張って慣れていきましょう😄

手順（Add＞Commit>Push）

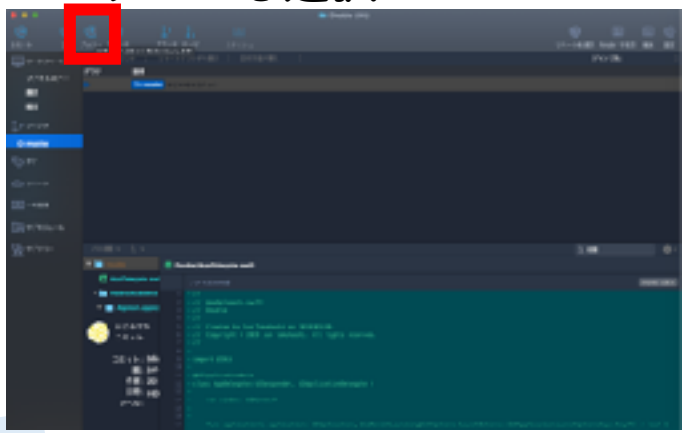
①作業ツリーのファイル
にチェックして次へ



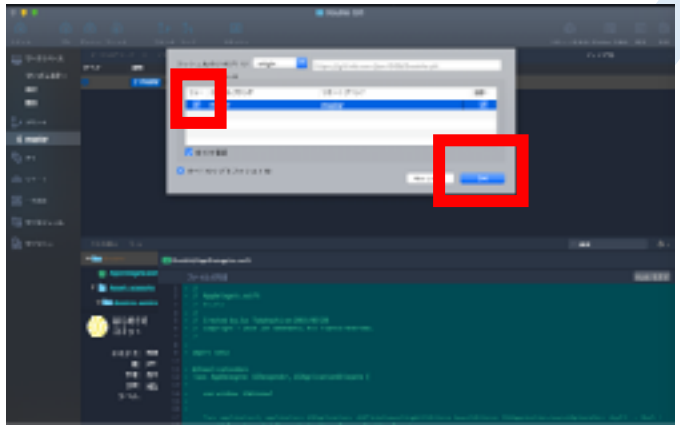
②コミットメッセージを
入力してコミット



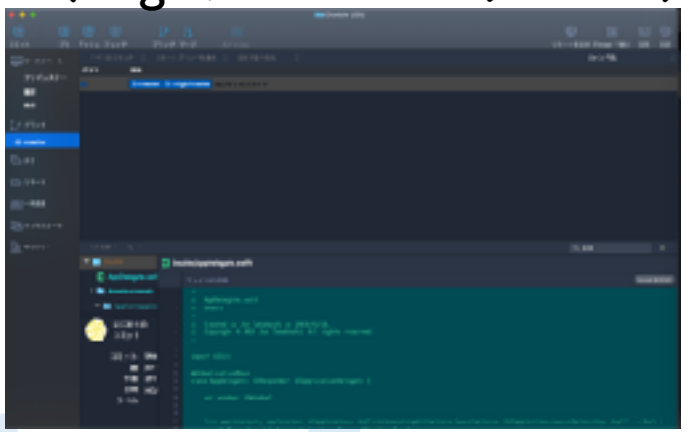
③コミットされたのを確認して
プッシュを選択



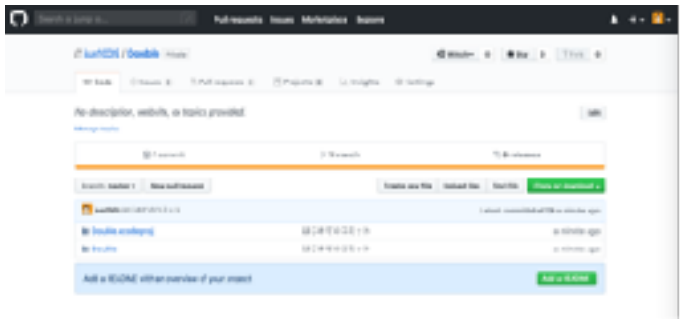
④ブランチ(master)を選択してOK



⑤プッシュされたことを確認
(origin/masterというラベル)



⑥GitHubにもあがっている！



ここまでのおさらい

こんなことを学びました

①Gitの概要について

②Gitコマンドについて

③iOSのプロジェクトをGit管理する手順（一例）

余裕があれば試してもらいたいこと

- 「ブランチ」を使う
- 「コンフリクト」の解消をやってみる

Gitの基本をもう少し学びたい人へ

https://backlog.com/ja/git-tutorial/

サルでもわかるGit入門

Gitの概要や基本的なコマンドなど、
こういった場合は？的な逆引きまで、
わかりやすく解説しているWebサイトです

意味不明だった人は、
復習がてらもう一度見ておくと◯

もっと学びたい人は
まずこの内容を理解できてから
Git関連の書籍にうつりましょう！