ノベルゲーム "想咲 - ソウサク -" の開発

開発チーム

- シナリオエディター開発チーム (〜5/22)
- ゲーム開発チーム
- テストチーム (関数テスト, 統合テストの実施)

開発スケジュール (予定)

```
2022年4月1日 〜 2022年7月31日
```

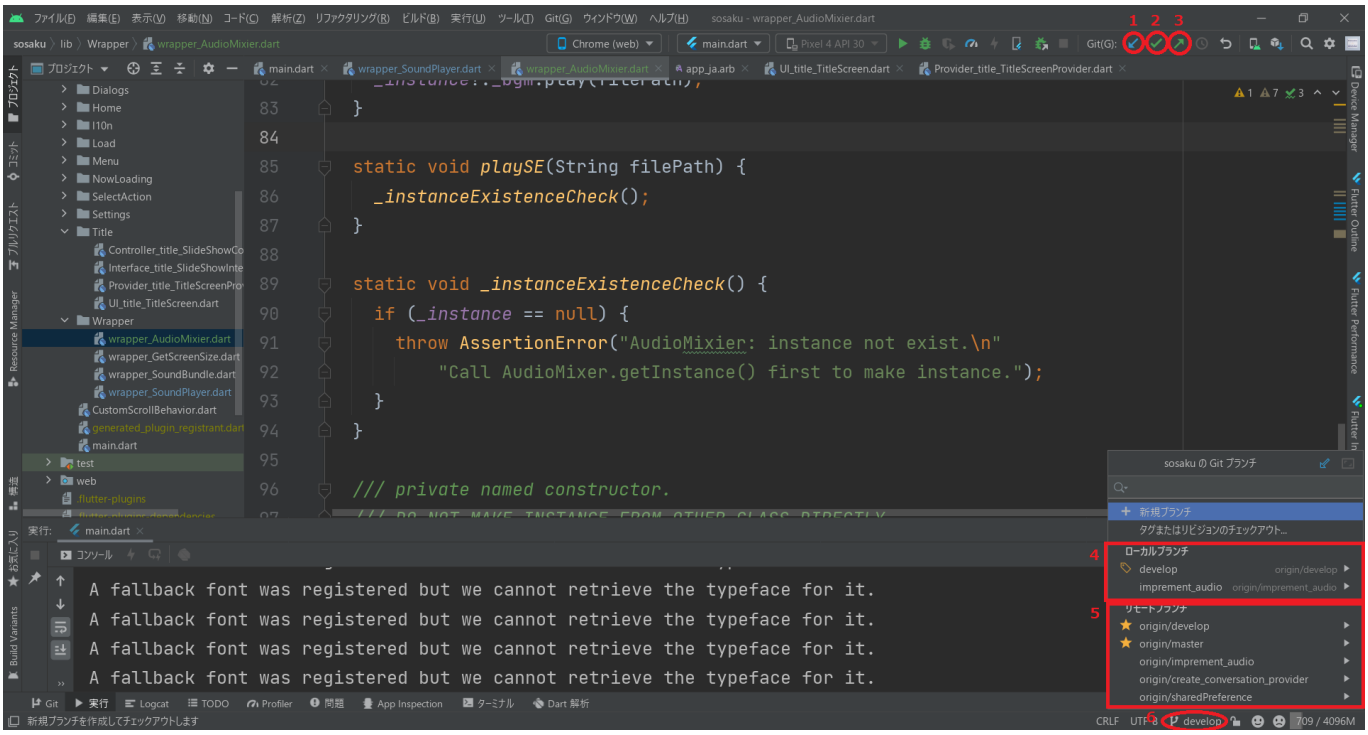
チェックポイント
2022/5/22 : シナリオエディター完成 -> シナリオ班へ渡す
2022/5/30 : UI完成 (表示内容などは未完成でOK)
2022/6/30 : 仮assetsでのゲーム完成
2022/7/15 : 美術班・音響班のassetsも取り入れた完成版完成
2022/7/31 : BugFix終了 -> AppStore, GooglePlay, w-sharp.comでの掲載

関係Webサイト

- 想咲 - ソウサク - GitHub リポジトリ
- シナリオエディター GitHub リポジトリ
- Lucid Chart

Gitについて

(Android Studio 2021.1.1 での操作方法なので他ソフト・バージョンの場合は適宜読み替えてください。)



1. プロジェクトを更新 (GitHubから最新のデータを取得してきてローカルを更新する)
2. コミット (今までの変更内容を保存する)
3. プッシュ (GitHubにアップロードする)
4. ローカルのブランチ一覧
5. リモートのブランチ一覧
6. 現在いるブランチ名

原則、作業を開始する前とブランチ移動後、マージ前にプロジェクトを更新してください
作業後、コミットとプッシュをするようにしてください。
コミット時のコメントは英語で書くのが吉。

作業が終了してブランチが不要になったら、ローカルとリモートのブランチを削除してください。

ブランチ構成

```
- master (最新のリリースビルドのデータ)　[消さないように!]
- develop (開発中の統合先ブランチ)　　[消さないように!]
- add_XXXX (各自が作成する新規ブランチ)　[終わったら消すように!]
```

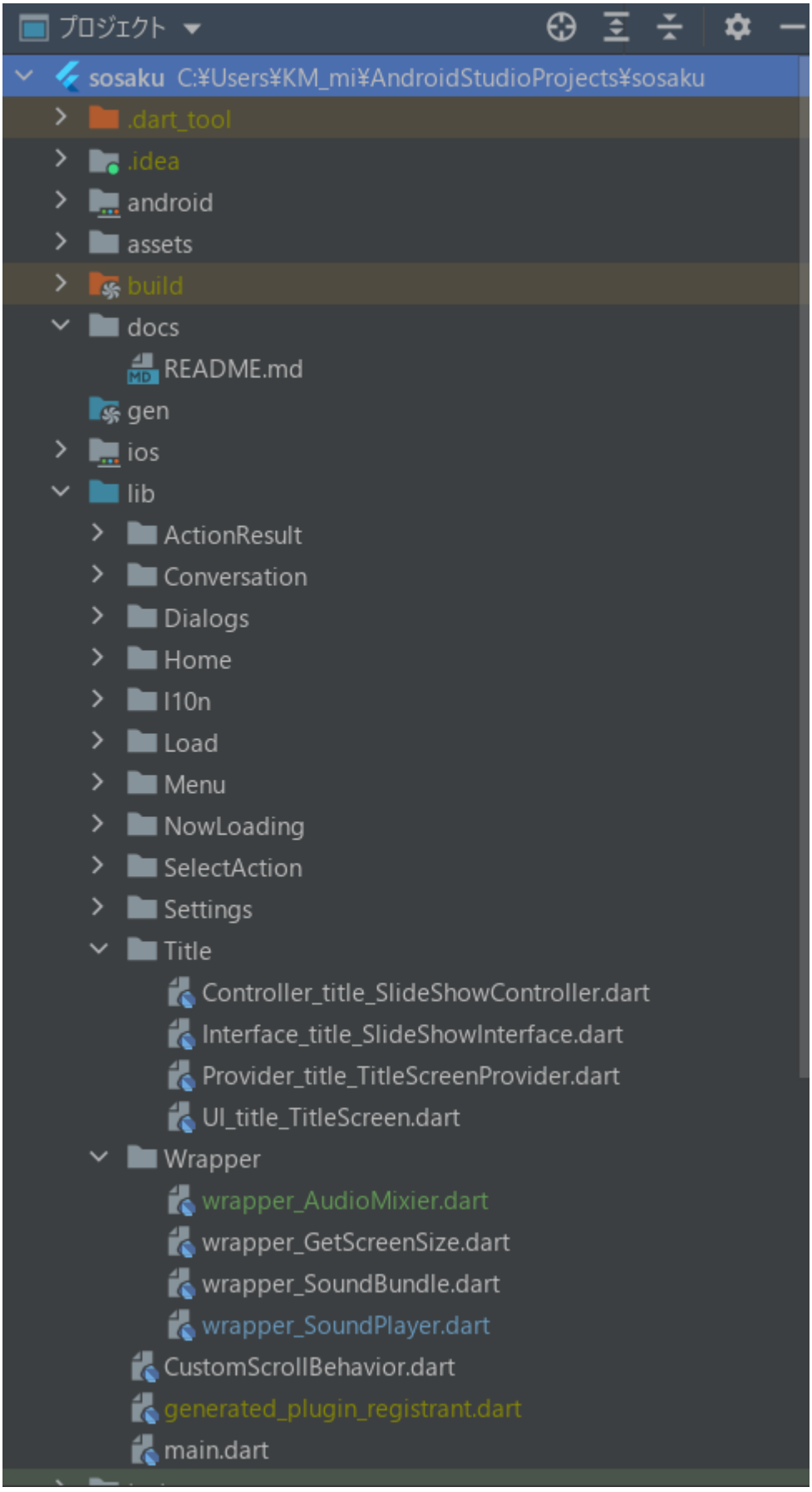
masterブランチはバクチェックなどが終わっている最新のリリースビルドを保存するブランチです。開発中には使いません。
developブランチは開発中の最新ビルドを保存するブランチです。動く状態が望ましいが、すぐに改善するなら動かなくても可。各自担当する機能の開発が終わったらここにマージしてください。

add_XXXブランチは各自が作成する新規ブランチを示しています。各自新しい機能の開発を担当したら、まずdevelopブランチから派生させて新規ブランチを作成し、それから開発を始めてください。 developで直接

開発するとほかの人のデータがとぶことがあるので厳禁。
名前は "add_XXX" や "fix_XXX" などメンバーが何の開発をしているブランチかわかるように付けてください。自分の名前などは禁止。

Gitやブランチについて分からない人は https://www.sejuku.net/blog/71071 などを参考にしてください。

ディレクトリ構成



- assets ディレクトリ : アセットが入っている
- docs ディレクトリ : ドキュメント関係が入っている

- lib ディレクトリ : dartのプログラムファイルが入っている

assets ディレクトリ

```
assets                  # すべての描画できるオブジェクト
├ drawable
│    ├ ActionResult
│    │
│    ├ CharacterImage   # キャラクターごとにフォルダー分け?
│    │
│    ├ Common           # UI要素など
│    │
│    ├ Conversation
│    │
│    ├ Dialogs
│    │
│    ├ Home
│    │
│    ├ Load
│    │
│    ├ Menu
│    │
│    ├ NowLoading
│    │
│    ├ SelectAction
│    │
│    ├ Settings
│    │
│    ├ Splash           # アプリ起動時のスプラッシュイメージ
│    │
│    └ Title            # スライドショーイメージ
│
├ sound                 # オーディオ関係
│    │
│    ├ BGM
│    │
│    ├ UI
│    │
│    └ Event
│        ├ X-X
│        │
│      (略)
│
├ font                  # フォントデータ
│
└ text                  # シナリオデータなど
```

lib ディレクトリ

```
lib
├─ ActionResult
│
├─ Conversation
│
├─ Dialogs
│
├─ Home
│
├─ Load
│
├─ Menu
│
├─ NowLoading
│
├─ SelectAction
│
├─ Settings
│
├─ Splash              # アプリ起動時のスプラッシュスクリーン
│
├─ Title
│
├─ Wrapper             # 複雑な処理を簡単に扱えるようにしたクラス群
│
├─ l10n                # localization support files
│   ├─ app_ja.arb      # UI要素(ボタンテキストなど)をここに書いて呼び出す。
│   ├─ l10n.dart       # 自動生成 (後述)
│   └─ l10n_ja.dart    # 自動生成 (後述)
│
└─ main.dart           # プログラムのエントリーポイント(アプリの各ライフサイクルでの処理はここに書く)
```

命名規則

- 変数名, 関数名は小文字から始めて単語ごとに大文字で区切る ex) audioVolume
- クラス名は大文字から始める ex) AudioVolumeController

libディレクトリ内のファイル名

- 原則対応する画面内のファイルに入れる。
- ファイル名は "<クラスタイプ>_<画面名>_<クラス名>"

ex) "Controller_conversation_ConversationScreenController.dart"

クラスタイプ

- UI : UIを定義するクラス (ConsumerWidget, StatelessWidgetなど)

- Provider : ChangeNotifierクラス
- Controller : ChangeNotifierに値を渡す、ゲームシステムを組み込んだ画面進行クラス
- Manager : どのControllerを呼び出すかを管理したり、状態をもとにControllerに指示を送る管理クラス
- Interface : implementsしたクラスに実装しなければならない関数の雛形を定義したabstractクラス

など

コメントの書き方

クラスのドキュメンテーションコメント

publicな変数、関数などにはコメントを残してください。

```
class TestClass {
  /// value for test.                    - 変数の説明
  int testValue = 0;

  /// Add two integers.                  - 関数の説明 (使用上の注意などがあれば複数行にして
書く)
  ///
  /// @param a : first integer.          - 引数の説明
  /// @param b : second integer.         - 引数の説明
  /// @return : sum of a and b.          - 戻り値の説明
  int addInt(int a, int b) {
    return a + b;
  }

  // 明日やる！                           - 普通のコメント
  /* int addDouble(double a, double b) { - コードのコメントアウト
    return a + b;
  } */
}
```

文字化け防止のため、コメントは英語で書きます。
"/// (スラッシュ3つ)"から始める書き方で書くと、Flutter Documentation Comment として扱われ、色が緑になって目立つとともに、他の人が呼び出すときに表示されます。

アノテーションコメント

自分がやることを列挙して置いたり、ほかの人に直してほしい部分があったり、まだ実装されていない機能についての注意だったりをコメントするときにエディターで目立たせることができます。

```
@override                                   - オーバーライドしたことを示すアノテーション
void test() {
  print('test');
}

// TODO: implement test class.              - やることを書いておく

// FIXME: Please fix test class.            - 修正する必要がある箇所に書く

// WARNING: addDouble() is not implemented. - 警告を書く
```

アノテーションコメントをすると赤や黄色に色が変わるので、長いコードの中でも探しやすくなります。
特にtodoはAndroid Studioの左下から一覧表示できるようになるので便利です。

詳細はこのページなどを参照してください。

L10nパッケージについて

UIのラベルを複数言語に対応させたり、ラベル名を1つのファイルで一元管理したりすることを目的にL10nパッケージを入れてあります。

書き込み方法

"lib/l10n/app_ja.arb" ファイル内に記述します。

```
{
  "@@locale": "ja",

  "gameName": "想咲 - ソウサク -",
  "@gameName": {
    "description": "このノベルゲームアプリの正式名称"
  },
  "tapToStart": "Tap to Start",
  "continueGame": "つづきから",
  "newGame": "はじめから",
  "settings": "設定",
  "gallery": "ギャラリー",
  "documents": "各種表記",
  "save": "セーブ",
  "backToHome": "ホームに戻る",
  "load" : "ロード"
}
```

以下のようにして項目を追加します。

```
"<キーの名前>": "<実際に表示されるラベル名>"
```

最後の項目にはカンマがつかないことに注意してください。

呼び出し方法

呼び出したいbuild関数内の任意の場所で、

```
L10n.of(context).<キーの名前>
```

を指定すると、コンパイル時に対応するString値に変換されます。

arbファイルのコンパイル

```
lib/Home/UI_home_HomeScreen.dart:98:65: Error: The getter 'settings' isn't defined
for the class 'L10n'.
 - 'L10n' is from 'package:sosaku/l10n/l10n.dart' ('lib/l10n/l10n.dart').
Try correcting the name to the name of an existing getter, or defining a getter or
field named 'settings'.
                    child: Button(buttonName: L10n.of(context)!.settings),
                                                                 ^^^^^^^^
Failed to compile application.
```

このようなエラーが出た場合には、Android Studioのターミナル画面で、

```
flutter gen-l10n
```

を入れてください。通常は Ctrl+S でセーブするときに自動的に実行されます。
この操作で、lib/l10n内の "l10n.dart" と "l10n_ja.dart" が更新されます。

flutter_native_splash パッケージについて

pubspec.yamlの一番下に、

```
flutter_native_splash:
  color: "#FFFFFF"
  image: assets/drawable/Splash/Wsharp.png
  fullscreen: true
  android_gravity: center
  ios_content_modes: center
```

の記述があります。ここを変更すると設定できます。
設定後は、Android Studioのターミナル画面で、

```
flutter pub run flutter_native_splash:create
```

を実行すると、各プラットフォーム用のスプラッシュイメージとその呼び出しコードが生成されます。