ISSN 1882-5370

尚美学園大学芸術情報研究　第 35 号

Journal of Informatics for Arts, Shobi University No.35

研究ノート｜Research Notes

アプリケーションの開発環境について

Development environment for application

須藤 智

SUDO Satoshi

尚美学園大学

芸術情報学部

Shobi University

2022 年 12 月

Dec.2022

アプリケーションの開発環境について

Development environment for application

須藤　智
SUDO Satoshi

[抄　録]
アプリケーションを開発する際に、最初にやるべき作業は開発環境の設定である。しかし、作成するデバイスや OS、アプリケーションによって様々な環境が存在し、設定すべきこともそれぞれ異なるため、何をすれば良いか分からず作業が止まってしまうことがよくある。スマートフォンのアプリでは Android と iOS とで制作方法が異なり、デスクトップアプリケーションにおいては、使用するプログラミング言語によって環境構築が大きく異なる。
そこで、本報告ではこれまでに学生が実際に卒業制作に使用した開発環境を中心に、作成するアプリケーション毎に開発に必要な環境やツールについて解説する。

キーワード
アプリケーション開発環境、統合開発環境、Android、iOS、Visual Studio、 Python

[Abstract]
When developing an application, the first task is to set up the development environment. However, since there are various environments depending on the device, OS, and application to be created, and the settings are different for each, work often comes to a halt because the developer does not know what to do. For smartphone applications, the production method differs between Android and iOS, and for desktop applications, the environment construction differs greatly depending on the programming language used.
In this report, I will explain the environment and tools necessary for the development of each application, focusing on the development environment that students have actually used for their graduation projects.

Keywords
development environment, Integrated Development Environment, Android, iOS, Visual Studio, Python

1. はじめに

2020 年度から小学校でのプログラミング教育が必修化され、プログラミング学習に関する様々なツールが開発されてきている。本学でも 1 年生からプログラミングの授業を配置し、卒業研究や卒業制作に必要なツールの開発やアプリケーション開発の基礎を学ぶことを目的としている。プログラミング言語は数多く存在し、制作する目的や環境によってどの言語を用いるかを決める必要がある。また、対象デバイス毎に開発環境が異なっており、アプリケーションを開発するための環境整備も初学者にとっては敷居が高い。

そこで、本報告では現在のアプリケーション開発の現状をまとめ、開発を始めるにあたって必要な手順をまとめる。

2. アプリケーション開発の基礎

アプリケーションを作成するためにはさまざまなプログラムや素材を作成し、すべてを組み合わせて一つのアプリケーションとして構成する必要がある。これらはもともと複数のツールを使用しそれぞれ作成していたが、1 つのソフトウェアとして使えるようにまとめたものを統合開発環境（IDE：Integrated Development Environment）と呼ぶ。開発用途に合わせて統合開発環境が用意され、プログラマーは対象デバイスや作成するアプリケーションによって使用する統合開発環境を選択する必要がある。

また、アプリケーションを開発するためには、特定の機能を提供するための「ライブラリ」と呼ばれるものがある。これらの多くはそれぞれの開発環境毎に用意され、アプリケーション開発時に組み込んで使用する。ライブラリにより、すべての機能を自分で作成する必要はなく、必要な機能を持ったライブラリを取り込むことで簡単にアプリーションを開発できるようになってきている。

これらのアプリケーション開発のための環境は基本的に無料で揃えることができ、書籍やインターネット記事を参考にして独学で学ぶことも可能である。実際にスマホ等のアプリを販売するためには登録等の作業が必要となり、登録料等が発生する場合もあるが、自分が所有している実機にて作成したアプリケーションの動作確認するところまでは無料で行える。

次章以降では、それぞれのデバイスでの開発環境についてまとめる。

3. スマートフォンアプリの開発

1990 年代後半から 2000 年代にかけて大きく普及した日本の携帯電話（通称：ガラケー）は通話機能がメインであるが、そのほかの機能としてメール送信や写真撮影なども可能であった。その後、2007 年に Apple 社から iPhone が発売され、その翌年 2008 年には Android 端末が発売され、現在では、これらのスマートフォンが個人で所有する通信デバイスの主流となっている。スマートフォンは指を使ってタッチパネルを操作するインターフェースを備えており、カメラでの写真撮影、GPS を使用した位置情報の取得、加速度センサーやジャイロスコープなどを利用して向いている方角やスマートフォンの傾き等の情報も取得することができる。アプリケーションも Android であれば Google Play から、iPhone であれば App Store からダウンロードして好きなものを使用することができ、また、個人でもアプリケーションを開発して販売することが可能となっている。このようにして、スマホの普及

によりアプリケーション開発は一般の人にも身近なものとなっている。ここでは、スマートフォンの多くを占める Android と iOS(iPhone の OS)の開発環境についてまとめる。

3．1　Android のアプリケーション開発

Android[1)]は、モバイル向けのオペレーティングシステム（OS）の名称である。主にスマートフォンやタブレットなどのタッチパネルを使用したモバイル向けが中心となるが、それ以外でもテレビ向け・自動車向け・ウェアラブルデバイス向けなども存在する。Android は Google 社が開発しているが、無償で誰にでも提供されるオープンソースであり、サードパティー（他のメーカー）のベンダーが独自にカスタマイズすることが可能であり、複数のメーカーから端末が発売されている。現在では Android 端末が世界中で多くのシェアを占めている。

Android のアプリを開発するための統合開発環境として、Google が提供する Android Studio がある。従来は、Eclipse という主に Java のアプリケーション開発に使用される統合開発環境に ADT(Android Development Tools)というプラグインを組み込んで使用していたが、2014 年にそれらが Android Studio のみで実現できるようになった。Android Studio は Windows や macOS、Linux など複数の OS で利用することが可能であり、Java 言語を用いて開発を行う。Java は多くのシステムやアプリケーション開発に用いられており、Android アプリの作成を通して Java 言語を習得すれば、さまざまなソフトウェア開発にも取り組むことができるようになる。最近は Java の代わりに Kotlin が使用されることも増えてきており、今後は Kotlin が主流になっていくと考えられる。図 1 に Android Studio を起動した画面を示す。

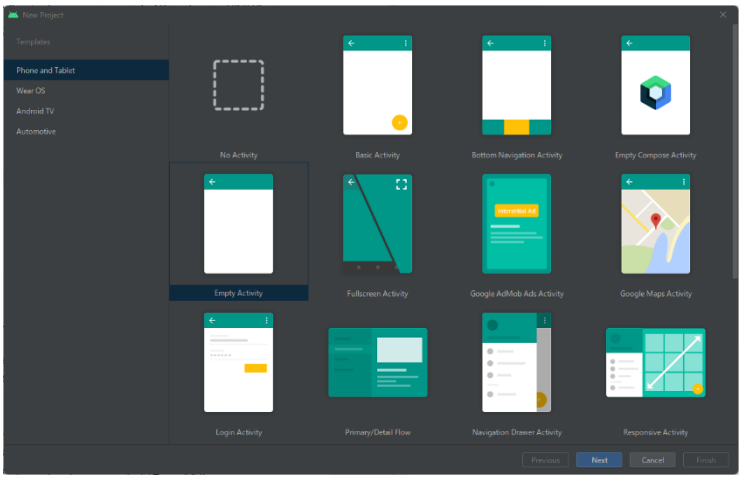

図 1　Android Studio のプロジェクト作成画面

Android Studio でアプリを開発するためには、まず始めにプロジェクトを作成する必要がある。アプリ開発に必要な設定等を自動で行い、基本的な画面やアイコンなどを配置した状態に設定してくれる。Android Studio を起動すると、図 1 のようなウィザードが表示され、作成するアプリのタイプを選択することで必要な設定ファイル等が自動生成され、プロジェクトが作成される。ウィザードに従ってプロジェクトを設定していくと、開発画面が起

動する。図 2 に実際の開発画面を示す。

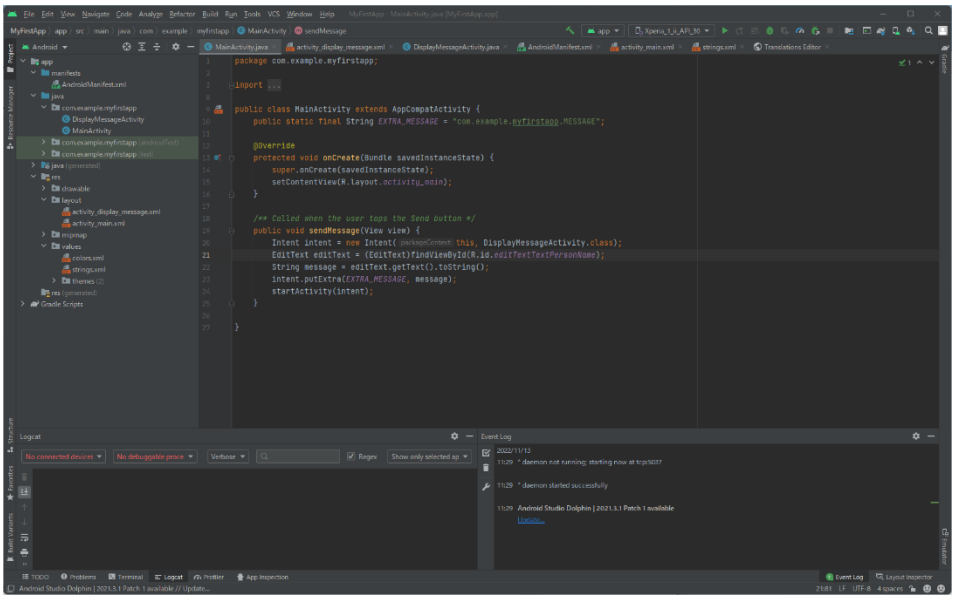

図 2　Android Studio の画面構成

統合開発環境は図 2 のように、ウィンドウがいくつかの領域に分割されている。画面で一番大きな領域が実際にプログラムを作成したりアプリケーションのレイアウト等を修正したりする編集画面である。それ以外にもプロジェクト全体のファイル構成をツリー状で表示したり、それぞれテスト実行時のログを表示したりするウィンドウもある。

このように Android のアプリケーション開発をするためには、Android Studio を使用し、Java や Kotlin による開発の修得が必要である。

3. 2　iOS のアプリケーション開発

iOS[2]は iPhone 向けのモバイル向けのオペレーティングシステム（OS）である。Android と違い、OS および端末の開発から販売までを Apple 社が行っている。

iOS のアプリを開発するための統合開発環境として、Apple 社が開発した Mac 専用の統合開発環境の Xcode がある。Xcode では、iPhone 以外にも Mac や iPad 向けのアプリケーションも開発できる。iOS は Swift と呼ばれるプログラミング言語を用いて開発される。これまでは Objective-C という言語を用いて開発されてきたが、2014 年に Apple 社が開発した Swift を使用しての開発が増えてきている。この Swift はスクリプト言語の設計思想も取り入れて開発されたことより、コーディングが簡単で直感的であると言われている。

図 3　Xcode の起動画面

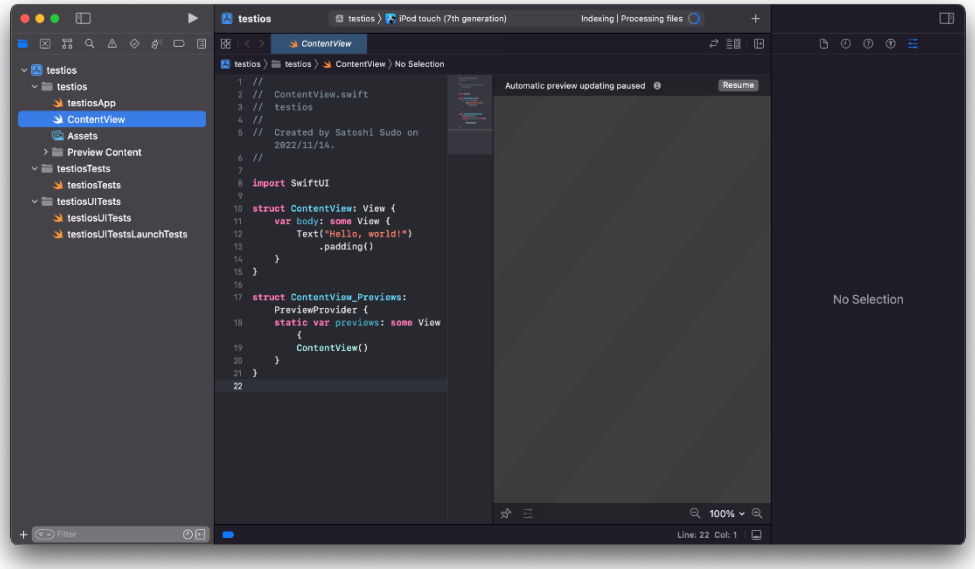

図 4　Xcode による開発画面

Xcode を起動すると図 3 のような起動画面が表示される。また、Xcode で iOS 用のプロジェクトを作成し、開発画面を起動すると図 4 のようになる。Android Studio と同様に、プロジェクト全体を管理するウィンドウやコーディングを行うウィンドウ、画面デザインを編集するウィンドウなどで構成されている。

このように、iOS のアプリケーション開発では、Xcode を使用して Swift による開発の修得が必要である。

3．3　Unity でのアプリケーション開発

これまでは Android と iOS それぞれの開発環境について現状を紹介してきた。それ以外に、Unity を使用してスマートフォンのアプリケーション開発を行うことも可能である。

Unity[3]とは、米国の Unity Technologies が提供するゲームエンジンであり、実際のゲーム開発でも多く使われている。簡単に 3 次元のゲーム開発が可能であり、物理エンジンも備えている。それ以外にも 2 次元のゲーム開発も可能であり、VR や AR などの仮想空間技術のプロジェクトも作成できる。このように Unity は基本的にはゲームエンジンでありながら、アプリケーション開発のための統合開発環境であるとも言える。

Unity は 3D キャラクターをプログラムを書かずに動かしたりすることができる。また、アセットストアと呼ばれる素材を管理する機能が充実し、簡単に 3D モデルをプログラム内で使用することができる。このように Unity は高機能なゲームエンジンでありながら簡単な操作で 3D のゲームを制作することが可能であり、プログラミング初心者にも取り組みやすいアプリケーション開発環境になっている。

Unity で開発したゲームは Windows・Mac などのデスクトップパソコンや、Android・iOS などのスマートフォンアプリなどでも動作可能であり、マルチプラットフォームでの開発に対応している。図 5 に Unity のプロジェクト作成時の画面を示す。

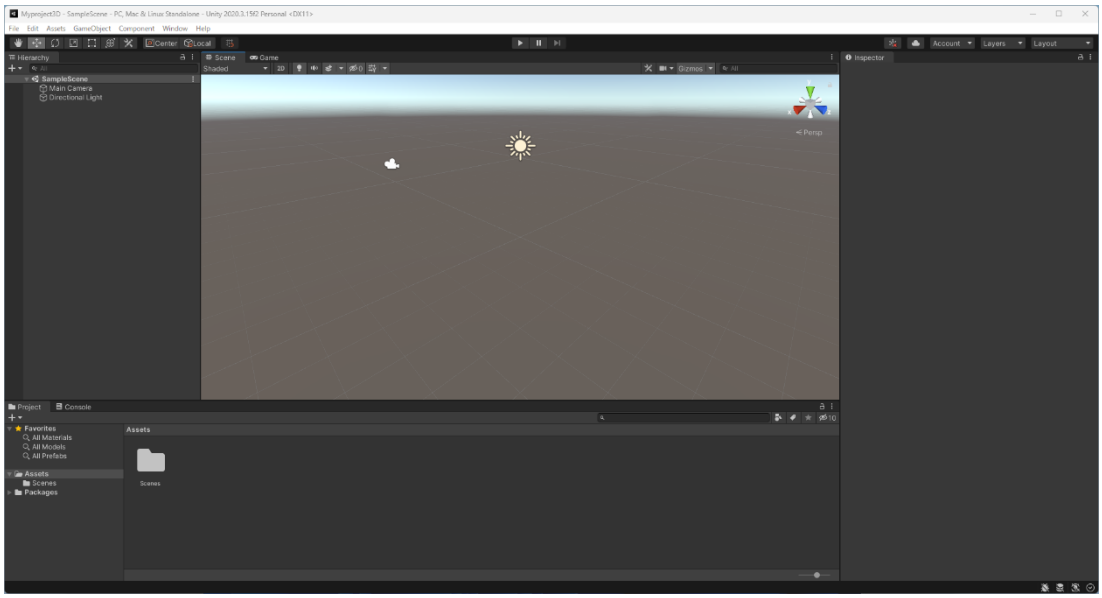

図 5　Unity の開発画面

3D のプロジェクトを作成すると、この画面のように既に光源やカメラが配置され、ここに 3D モデルを配置し、動作を設定することで簡単に 3D キャラクターを動かしたりすることができる。

Unity は複雑なプログラミングを書かなくても、グラフィカルな操作でアプリケーション開発が可能なため、初心者でも比較的取り組みやすく、これまでも卒業制作にて使用する学生は多い。

4. デスクトップアプリケーションの開発

スマートフォンアプリと異なり、コンピュータのデスクトップ環境上で動作するアプリケーションのことをデスクトップアプリケーションと呼ぶ。多くの開発方法が存在するが、これまでの研究室での卒業制作での使用環境を中心にまとめる。

4. 1　Visual Studio での Windows デスクトップアプリケーションの開発

Windows でのアプリケーション開発の多くは Microsoft 社の Visual Studio[4)]を使用して行われている。Visual Studio では、Windows でのアプリケーション開発はもちろんのこと、デバイスドライバーやサービス、Web アプリなどの開発も可能である。また使用する言語も、C 言語や C++のようにネイティブコードを生成する言語から、C#や.NET のようにマネージコードを生成する言語などのコーディングやデバッグをサポートしている。メジャーバージョンアップも 3〜5 年毎に行われ、新しい機能などが追加されている。また、通常は有償ではあるが、個人開発者向けの Community というエディションが存在し、条件を満たすユーザは有償版の Professional エディション相当の機能を無償で利用することが可能

であり、学生もこのエディションにてプログラム開発を行うことが可能である。Visual Studio の最新版は 2022 であり、初の 64 ビットでの IDE となった。Visual Studio を起動したときの画面を図 6 に示す。

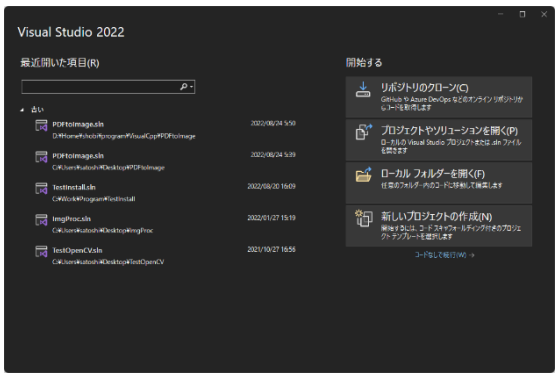

図 6　Visual Studio 起動時の画面

図 7　新しいプロジェクトの作成

また、起動画面にて「新しいプロジェクトの作成」を選択すると、図 7 のように使用可能なプロジェクトのテンプレートが表示され、作成するアプリケーションの種類を選び、ウィザードに従って設定を進めるとプロジェクトが作成される。また、プロジェクトを作成した後の Visual Studio の画面は図 8 のとおりである。

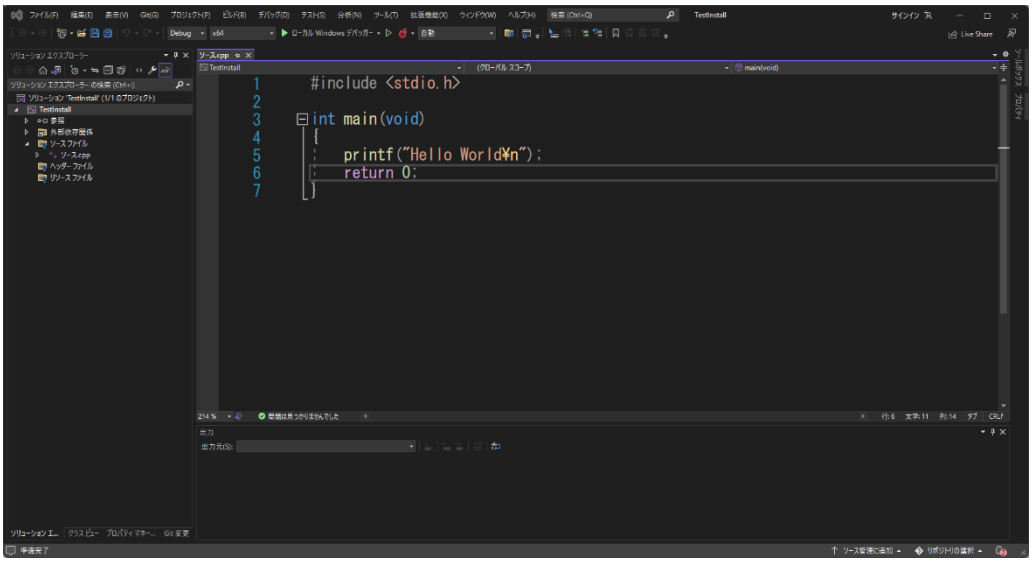

図 8　Visual Studio での開発画面

これまでの統合開発環境と同様に、複数のウィンドウに分割され、コーディングするエディタ部分、構造情報を提示する部分や実行やコンパイル時のメッセージを提示する部分がある。

この環境でのアプリケーション開発の例として、「C++言語」にて「DX ライブラリ」を利用したゲーム開発が行われてきた。通常、Windows のアプリケーションではマルチメディアに関する機能実装には「Direct X」と呼ばれるゲーム・マルチメディア処理用の API の

ライブラリを使用する。しかし、プログラミングの経験が浅い学習者がこのライブラリを使用するためには記述が煩雑であり、理解できない箇所も多く、プログラミングの学習の環境には向かない。そこで、この「DirectX」を初学者でも簡単に使えるようにラッピングした「DX ライブラリ」を使用して、これらの機能を簡単に利用することができる。本ゼミでは、「DX ライブラリ」を使用してゲーム開発を進めることを授業の一環として行っており、ここで Visual Studio の使い方も学習している。

4. 2 Python

Python[5]は汎用的なプログラミング言語で、システム管理・アプリケーション開発・科学技術計算・Web システムなどで広く利用されている。特に、2010 年代ごろからの機械学習ブームでは、大きな注目を集めており、機械学習を行うパッケージは Python を利用したシステム構成となっている。また、Python は多くの開発者がライブラリの開発を行っており、これらを利用することで様々な機能を持ったアプリケーションを簡単に作成することが可能となっている。

Python をインストールする方法は 2 種類存在する。1 つは公式版の Python をインストールする方法で、もう一つは Anaconda 版を使用する方法である。一般的なプログラミングの学習や開発では多くのライブラリを使用する必要は無く、必要なライブラリのみをインストールして使用することにより基本的な構造も理解できるため、公式版の Python を利用する方が良いと考えられる。一方、Anaconda は機械学習や統計処理などさまざまな科学技術計算のためのプラットフォームで、その中心的なツールとして Python を配布している。AI やディープラーニングなどを実装するためには Anaconda 版をインストールすると良い。

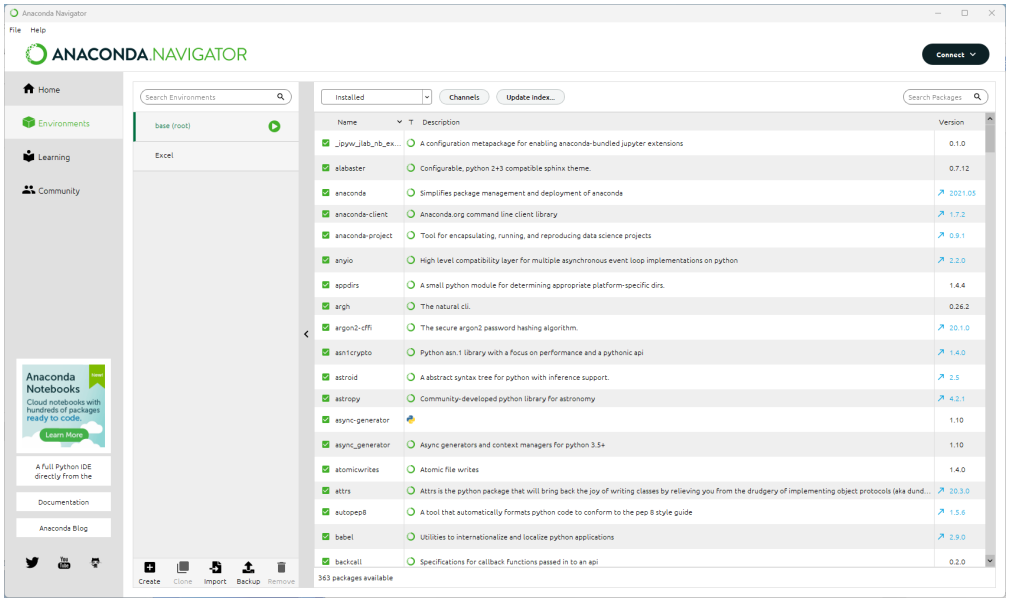

図 9 Anaconda Navigator

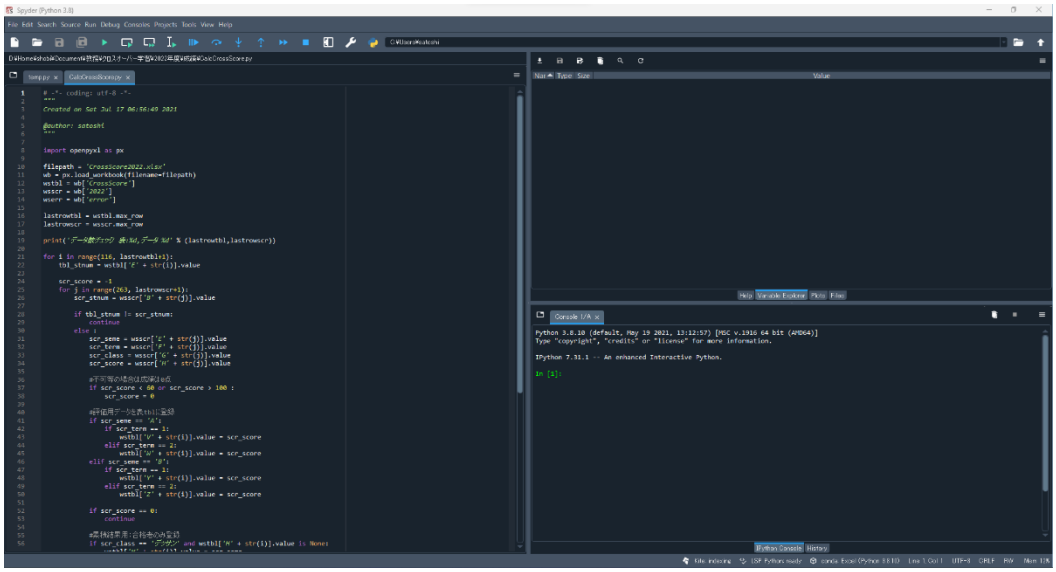

図 10　Spyder での開発画面

Anaconda では、図 9 に示すパッケージや環境を管理する Anaconda Navigator を利用して環境を整備していく。起動すれば分かるとおり、既に多くのパッケージがインストール済みで、これらを使ってすぐに開発を始めることができる。また、Python の開発では、仮想環境の管理も重要である。Python ではプロジェクトごとに仮想環境という独立した実行環境を作成しその中で開発を行う。インストールしたパッケージはお互いに参照しあっており、あるパッケージのみをアップデートすると、それを参照しているパッケージで不具合を起こす可能性があるなどバージョン管理が難しい。そこで、仮想環境という独立した実行環境を作成し、その中でパッケージのインストールや更新等を行い、開発環境を整えてアプリケーションの開発を行う。Anaconda Navigator はこの仮想環境の管理も行える。

Anaconda には Spyder と呼ばれる Python 向けの統合開発環境もあらかじめ入っている。起動すると図 10 のような画面構成となる。Spyder も他の多くの統合開発環境と同様に、プログラムを記述する編集画面、実行結果を表示する出力画面などに分割されている。Anaconda はこのように、Python 本体と多くのパッケージ、統合開発環境などのツールがインストールされており、すぐにアプリケーション開発を始めることができる。そこで、Anaconda を使用して Python の環境を整備し、アプリケーション開発を行っていることが多い。

5. 具体的なアプリケーション開発の例

ここでは筆者が開発したアプリケーションの開発環境についてまとめる。今回紹介するアプリケーションは「パソコンによるドローンの飛行制御」を行うものである。制御するドローンは RYZE 社の「Tello」[6]である。機体は重量約 80 g と軽量で誰でも簡単な操作で飛ばすことができる。Tello は有名なドローンメーカーDJI 社製のフライトコントロール技術を搭載しており、その場にとどまる「ホバリング」は非常に安定し、離着陸も簡単な操作で自動で行う。Tello はプログラムでドローンを操作することも可能であり、Python で操作するための SDK が公開されている。ドローンとの通信は Wi-Fi 接続で行う。通常、スマホ等

のアプリで Tello を操作する場合も同様で、Tello が Wi-fi の親機となって、ルーター機能を持つ。スマホが Tello のネットワークに接続し、ネットワーク通信によりコマンドを送信したり、データを受信したりすることで通信が可能となる。パソコンと Tello を接続する方法も同様で、Tello の Wi-fi ルーターにパソコンから Wi-fi で接続を行い、プログラムからコマンドを送信すれば制御でき、データを受信してドローンの情報を受け取れる。また、Tello には EDU というバージョンも存在し、こちらは複数のドローンが編成を組んで飛行するドローン編隊が可能である。この場合、Tello の他に Wi-fi ルーターを用意しそれを親機とする。Tello の各機は先ほどとは異なり Wi-fi 子機としてルーターに接続することにより、複数台の Tello が同一ネットワーク上に接続することができる。また、Tello を制御するためのパソコンも同一ネットワークに接続し、パソコンから複数の Tello へネットワーク内の通信にてコマンド送信が可能となる。ただし、事前に各 Tello の IP アドレスの情報が必要であり、コマンド送信は IP アドレスを指定して行う。パソコンでの開発環境は Anaconda により Python にて行う。Tello の Python 用の SDK はソケット通信により命令コマンドを送信したり、機体のステータス情報を取得したりするので、特別なライブラリのインストールは必要無いが、簡単に SDK を使用できるライブラリ「tellopy」を追加でインストールした。また、Tello のカメラ画像を取得し、画像処理を行うために、Python 用の OpenCV モジュールを追加し、Tello のカメラ映像を画像解析する。このような環境により Tello をパソコンで飛行制御し、Tello からのカメラ映像を画像解析する環境が構築できる。

6. むすび

スマートフォンの普及によりアプリケーション開発はより身近なものになった。開発環境を無料で揃えることができ、インターネット上で開発の方法を詳細に説明しているページもあり、一般の人でもパソコンさえあればアプリケーションを開発し、公開・販売することが可能である。

本学に入学してくる学生は卒業制作でアプリケーション開発を志望している学生もいて、1 年生からプログラミングの授業を受けている。プログラミングは多くのプログラムを自分で作ってみて、様々な環境でのプログラミングを経験することが大切だと考えている。そこで、アプリケーション開発を志す学生には、自分の興味のある分野でのアプリケーション開発にチャレンジして欲しい。そのためには、開発環境の整備が必須であるため、本報告を参考にどのようにすればアプリケーション開発ができるのかを調べながら環境設定してみると良い。いろいろと試行錯誤して環境を構築し、自分が作成したプログラムが正常に動作したときの達成感は今後のアプリケーション開発のモチベーションとなる。

これまでゼミ生の卒業制作を指導してきたが、アプリケーション開発で一番躓いているのは開発環境の導入である。これは各自が自分のノート PC にインストールし、環境設定を行う必要があるため、うまく動かないときの対処法が人（ノート PC）それぞれ違っている。そこで、早い段階から本報告を参考に様々な開発環境を導入し、アプリケーション開発に挑戦すれば、実際に卒業制作で開発を行う際にも慌てずに準備ができると思う。

引用文献・URL

1） https://developer.android.com/studio/intro?hl=ja
2） https://developer.apple.com/jp/xcode/
3） https://unity.com/ja
4） https://visualstudio.microsoft.com/ja/
5） https://www.anaconda.com/
6） https://www.ryzerobotics.com/jp/tello