OS 自作班活動報告書

立命館コンピュータクラブ 2016 年度 前期グループ活動

OS 自作班
海木 一佳 *
山田 知葉 †
安田 直樹 ‡
福井 尚卿 §
安田 弘樹 ¶
平原 滉大 ‖
広田 大地 **

2016 年 11 月 15 日

* 情報理工学部 二回生
† 理工学部　三回生
‡ 情報理工学部 二回生
§ 情報理工学部 二回生
¶ 情報理工学部 一回生
‖ 経済学部 一回生
** 情報理工学部 一回生

目次

文責：海木 一佳

1　要旨

我々の班は Windows や Linux には遠く及ばないものの一般的な PC に搭載されるような機能に近い機能を持った OS の制作を考えた．最も身近であり，普段使用している OS がどのような動作処理をしているか非常に興味があったからだ．開発にあたって，初めから作ることは困難なので「30 日できる OS 自作入門」という書籍 (以下，OS 本とする) を用い，OS に対する理解を深めるとともにオリジナル要素を持った OS の自作を目指した．ただし，OS 本は Windows 環境での開発が前提とされており，Mac ユーザーにとって開発がとても困難なものとなっていた．そのため，Mac ユーザーは OS 本ではなく，OS に関することに限定したうえで，各自自由に調べ学習と制作をしてもらった．

制作した OS の想定動作環境はプロセッサエミュレータである Qemu である．OS 本では Qemu とフロッピーディスクからの起動が想定されているが，フロッピーディスクからの起動は検証していない．フロッピーの入手と読み込みが手間であり，そもそもフロッピーを使用すること自体が時代錯誤であると考えたためである．開発環境は Windows，使用言語は C 言語とアセンブリ言語である．主に使用されているのは C 言語ではあるが，C 言語ではカバーしきれない部分があり，アセンブリ言語によってその部分を補っている．

2　搭載機能・アルゴリズム

2.1　割り込み処理

作成した OS はマウスやキーボード，マルチタスクに対応している．対応にあたって，割り込み処理を導入することで効率的かつ素早い処理を実現した．割り込みは PIC(programable interrupt contrillor) と呼ばれるチップセットが集積し，CPU に伝える．PIC の各レジスタを設定することで割り込み処理ができるようになるのだが，C 言語では PIC のレジスタに直接値を設定することができない．そのため，アセンブリ言語で代入する関数を制作し，その関数を用いることで対応している．割り込み処理を実装するにあたり，処理の高速化は非常に気を使った．割り込み禁止の時間の分だけ，他の処理が遅れてしまうからである．if() 文や for() 文は時間のかかる処理なので、できるだけ減らすように努めた．高速化を進める上で，他にも番兵法やリスト構造などを用いた．

2.2　画面表示

本項では，画面表示の手法について説明する．OS の画面表示には VRAM を用いる．VRAM は「Video RAM」の略称で，各番地が画面上の画素に対応しているため，この値を書き換えることで画面に任意の情報を出力する事ができる．開始番地:p 画面の横幅:sx とすると，任意の座標 [x，y] に対応する VRAM の番地は p+x+y*sx この番地の値を変更すると，その座標の色を変更することができる．

■図形の描画　ウィンドウ等を描画するために，長方形を描画する処理を実装した．左上頂点 [x，y]，横幅 [w]，高さ [h] を描画する処理は p+x+y*sx から p+(x+w)+(y+h)*sx まで w，h の値をループさせて，全ての座標に色を入力することで長方形を描画することができる．

■文字の描画　文字を描画する場合，フォントデータを作成する．各ドットに対して，描画するか，しないかの情報を持つフォントを，使用する文字の種類分用意してその情報を基に描画する．以下がフォントデータの例となる．

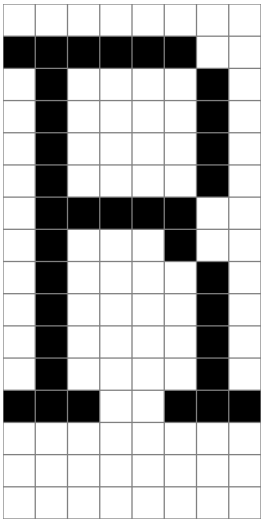

図 1　フォントデータ

■重ね合わせ処理　複数のデータを描画する場合，レイヤーを用いることで，効率的に描画することができる．例えば，普段使用している OS では，バックグラウンド，アイコン，エクスプローラ，マウスカーソルなど，様々な情報を描画している．これを，描画対象毎に，大きさと，描画の優先順位を設定することで，上下関係を明確に，かつ描画内容を最小限に抑えて描画することができる．

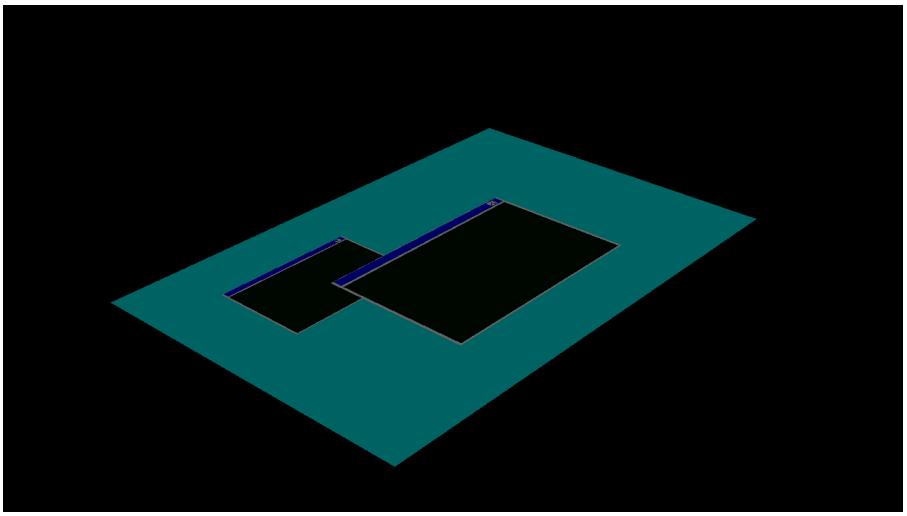

図 2　重ね合わせ図 1

この様に 2 つのウィンドウと背景が描画された画面がある場合，

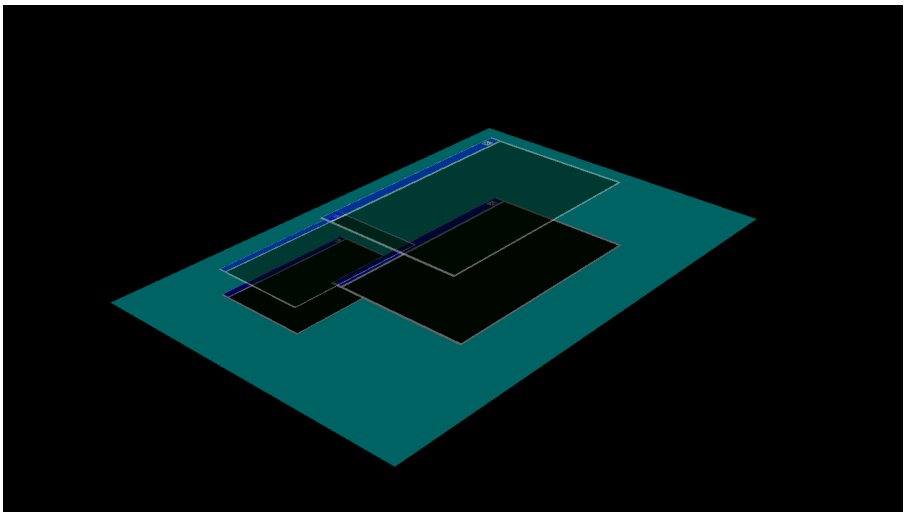

図 3　重ね合わせ図 2

背景レイヤー，ウィンドウレイヤー 1，ウィンドウレイヤー 2 に分割することができる．

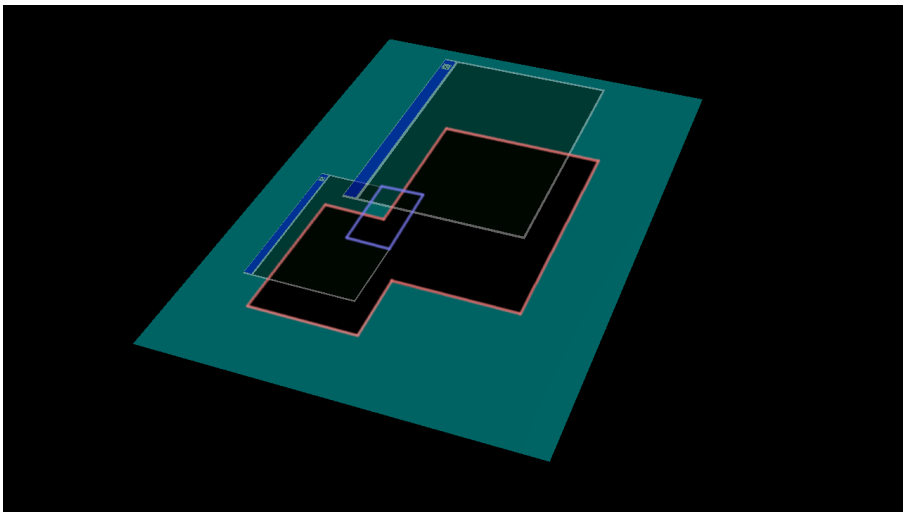

図４　重ね合わせ図３

この時，上のレイヤーと重なっている部分は描画する必要がないので，描画を行わないことで高速化できる．

2.3　マウス制御

ＯＳでマウスを使うために「ＣＰＵへマウスに関する割り込みを発生させる為にマウス，及び基盤に設けられたマウスの制御回路へ有効化命令を下して作動させる関数」，「マウスから送られる信号を数値化・それを処理する関数」，「目に見える形で関数の結果を表す物（マウスポインタ）」を制作した．マウスが一度に送る情報は「Ｘ軸移動・Ｙ軸移動・キー入力状態」の３バイト分である．この三つの情報は同時に入力されないといけないが，マウスからはこの情報群が同時ではなく，決まった順番でひとつずつ送られてくる．そのため，バッファを用いる事で送られてきた情報を１バイト分ずつ記憶，情報が三つとも揃い次第同時に入力する事で処理している．また，マウスポインタはただ重ね合わせ処理を行って描写するだけでなく，他の画像と重なった際にはどの画像よりも手前にマウスポインタを表示させたり，後ろに隠れている画像の更新を停止させたりする事でマウスポインタのちらつきを防ぐなど，マウスポインタの動きを分かりやすくするための関数も幾つか用意している．

2.4　メモリ管理

メモリの容量によって実行できるプログラムやデータの大きさが決まる．メモリを効率よく利用するため，OS におけるメモリ管理は重要である．メモリ管理では主に必要なメモリの確保と，使い終わって不要になったメモリの解放を行う．

この章ではメモリ管理の方法として，セグメント方式とページング方式の二つを挙げる．

メモリ容量チェック

はじめに，メモリ管理を行うためにメモリの容量を確認する必要がある．メモリチェックは CPU 内のキャッシュ機能（CPU 内の演算回路とメモリの間にある）を一時的に無効化し，メモリに適当な値を書き込み，その直後にそこを読み込んでから書き込んだ値と等しいかどうかで判断している．キャッシュを無効化する理由はキャッシュはメモリとメモリに読み書きした値を記憶してしまうため，直接メモリに読み書きせずにキャッシュから値を読み書きしてしまうためである．

実際のメモリチェックの作業は，star t 番地から end 番地までの範囲をポインタ*p で指定し，変数 old でメモリの元の値を保存する．そして 0xaa55aa55 を試しに書いてから反転させて結果が正しいか比較を行う．

セグメント方式

4GB の仮想メモリをセグメントという可変長のブロックに分割，メモリを確保し，それらを組み合わせてメモリを管理する．

実際の作業では，メモリに 0 と 1 を書きこんで，どこが空いている領域なのかを for 文で探す．例えば，4KB 単位で管理して 100KB の領域が欲しいならば 0 が 25 個並ぶところを探す．メモリを確保する場合は 1 を入れ，解放する場合は 0 を入れる．

セグメント方式での問題として，各セグメント間に空き容量が点在してしまう断片化が挙げられる．

ページング方式

効率的なメモリの利用方法として，多くの OS で採用されている方式である．4GB の仮想メモリを固定長のページという連続したメモリのブロックで分割，メモリを確保するメモリ管理法である．ページという比較的小さな単位のメモリを割り当てるため，断片化の問題を効率的に解決できる．

3　オリジナル改変

3.1　エディタらしきもの

14 日目で作ったウインドウを改変し，保存機能はないがエディタのようなものを制作した．改行機能，デリート機能，文字の色変え機能を備えている．キーボードで C を入力する度に色が変わり，B を入力することで黒に戻すことができる．ウインドウ本体も書籍にあったものから大きくしたが，あまり大きくすると画面描画の処理が追い付かず，チラつきや遅延が発生した．そのため 11 行× 17 列の大きさにしている．列の高さを管理する変数を作り，改行は管理した．

図 5　エディタ

3.2　ライフゲーム

これまでで得られた知識を活用し，開発した OS を独自に改変した．今回は，開発した OS の中で実行可能なアプリケーションを作成した．

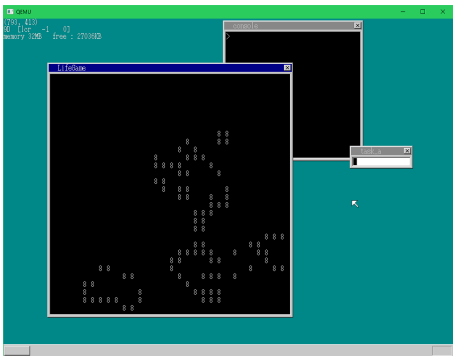

図 6　ライフゲーム

アプリケーションの内容は，ウィンドウ内で「ライフゲーム」を行うものとなっている．「30 日でできる！ OS 自作入門」17 日目に作成するコンソールを改変して作成した ADSW キーでカーソルを移動， Z キーでセルの状態変更，N キーで次の世代へ移動する．

4　Mac 班

4.1　マルチ OS 向けアプリケーションテスト支援ソフトウェア

Node．js のインストール

　このソフトウェアは Node．js によって Run とネイティブアプリへのビルドが作られているため以下の URL から Node．js をインストール．
https://nodejs．org/en/

ネイティブアプリへの書き出し環境作成

　このソフトウェアは，Adobe 社が公開している Phonegap というフレームワークを使うことにネイティブアプリへの書き出しを行っている．
そのため，Phonegap のインストールが必要である．
Phonegap は Node Package Manager を使って PC にインストールできる．
Node Package Manager は Node．js をインストールすると自動でインストールされる．
以下のコマンドで Node Package Manager を使って Phonegap をインストールできる．

```
「Terminal」
> sudo npm install -g phonegap
```

ソフトの Run

　このソフトウェアは Node．js を使って動かすことができる．
先ほど解凍したディレクトリに行き，Terminarl で下記のコマンドを実行する

```
「Terminal」
> cd ディレクトリ名
> node app
```

ソフトを使う

　適当なウェブブラウザ (Chrome が好ましい) を開き，下記の URL にアクセスする．
http://localhost:7000/

ソフトの終了

　ウェブブラウザを閉じても Node が起動したままになるので，Terminal にて Control + c を押して Node を終了する．

```
「Terminal」
> node app
node> 「Control + c」
```

>

ソフトを使ってアプリケーション開発をする

ソフトウェアのディレクトリー内の構造は下記のようになっている．

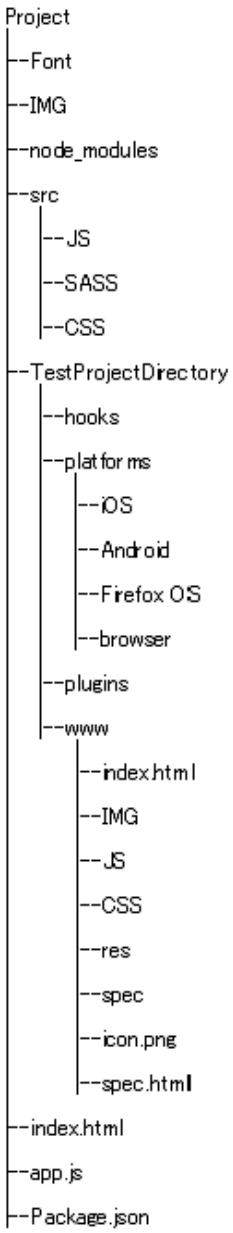

図 7　ディレクター内構造

通常，開発をする際にいじる必要があるのは．/Project/TestProjectDirectory のみである．

この中のディレクトリの説明をすると，

TestProjectDirectory/CSS このディレクトリ内に CSS を書く

TestProjectDirectory/JS このディレクトリ内に JS を書く

TestProjectDirectory/JS このディレクトリ内に JS を書く

TestProjectDirectory/index．html このファイルが起動に立ち上がる初期画面となる．

なお，このソフトを起動していれば赤枠内のボタンを押すことによって現状をリアルタイムで見ることができる．

図 8　起動画面

作ったアプリケーションをネイティブアプリとして書き出す

Terminal でソフトのディレクトリを開き以下のコマンドを入力

```
-Android
「Terminal」
> npm run build-android

-iOS
「Terminal」
> npm run build-ios

-FirefoxOS
「Terminal」
> npm run build-firefox

-WindowsPhone
「Terminal」
> npm run build-WindowsPhone
```

書き出されたネイティブアプリは TestProjectDirectory/platforms に保存される．
保存されたソフトウェアは iOS なら Xcode，Android なら Android Studio から編集できる．
なお，再度書き出しすると上書き保存される．

5　まとめ・今後の展望

最終的に本の進捗は 15〜20 日目分までとなった．基本的に週に 3 日分くらいを目安に進めていたが，別件で忙しい人も多く，できない人が多かった．また，テスト期間により 3 週間近く活動を停止したこともあり，期間中に完遂

することは難しかったと考えられる．16 日目を終了させた時点でシステム面は大方完成しているため，オリジナル要素の模索を優先した．完遂こそできなかったものの，OS に関して深い知識を得られた．今後の展望として，まずは完遂できなかった OS 本を完遂させたい．それから得られた知識を生かしさらなる発展を目指していく．