んけうす

数学研究部　平成二十一年度活動報告

-目次-

はじめに

本日は数学研究部の展示においでいただき有難うございました。活動内容の発表の場としてこの小冊子「すうけん」を発行しています。数学研究部は現在はプログラミングを中心に活動しています。プログラミングといってもほとんどがゲームを作ることになるのですが、ゲーム制作を通して、数学的な考え方やコンピュータサイエンスの入り口を学び、そして何よりも、「ものを作る楽しさ」を感じてくれたらと考えています。

今年は昔懐かしいパズルゲームとシューティングになりました。それぞれのゲームの中で使ったアルゴリズムについて説明をしています。新しい試みとして、「THE HISTORY OF COMPUTER CHESS」の翻訳と「コマンドプロンプトについて」の解説が入りました。内容をご一読いただければ幸いです。

SameMutch

2009/11/3(Tue)　齊藤　亘

1、概要

入部当時はプログラムに関して全く知識がありませんでしたが、漠然と「ゲームを作ってみたい」と思っていました。そこで去年の SF では RPG ツクールというゲームソフトで RPG ゲームを作りました。あれほど馬鹿らしいゲームを作った自分を責める気持ちでいっぱいです。あのゲームは相当くだらないし内容も無いのに、異様に容量を食うので今すぐアンインストールすることをおすすめします。

C++を死に物狂いで勉強して、ついに C++の凄さが理解できました。まだまだ未熟ですが、なんとか今回のゲームが出来ました。今年こそは**配布ではなく販売したい**、と何度も言っていましたが顧問の鶴の一声で配布に決定しました。その原因は何といっても発表できる作品の少なさです。今回発表できるアプリケーションはたったの二つ（顧問の作品を除く）。高校一年が三人もいながらプログラムが出来るのは自分だけ（一人は入部間もないのでしかたがないが、もう一人は・・・）。この恥ずべき状況を何とか打開したいと思っています。

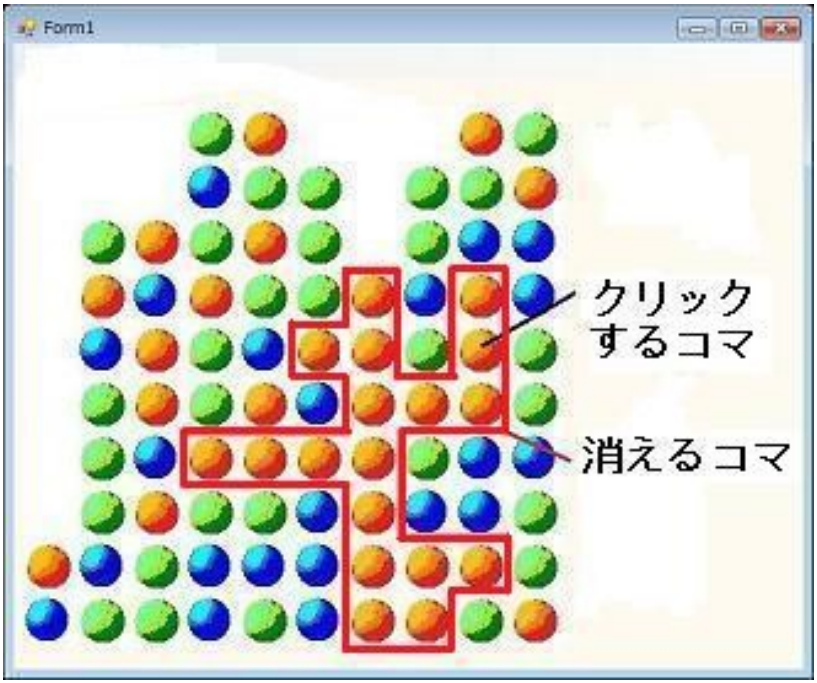
来年の SF までの目標は、

部員全員がプログラムが出来るようになることと、**販売する**ことです。後輩の方はやる気があって宜しいのですが、高校一年の方が暇さえあればサボろうという悪い雰囲気です。これからの課題は後輩の面倒を見る、というよりも同級生の面倒を見ることのようです。

2、遊び方

画面上のコマを選んで消していき、得点を競うゲームです。

同じ色が 2 つ以上タテ・ヨコに隣接しているコマは、選んで消すことができます（孤立しているコマは消せません）。



得点は(消えるコマの個数－2)の二乗が得点になります。つまり同時に消えるコマの数が多いほど高い得点になります。
消えたコマの上にあったコマはスムーズに流れるように下に落ちてきます。
縦一列のコマが全て消えると、その右隣の列は左に移動します。
消せるコマが無くなった時点でゲーム終了です。
全てのコマが消えるとボーナスとして500点が入り、得点を引継いで次のステージに進むことが出来ます。

3、その他

タブメニューの「ゲーム」から、
「リセット」で新しくゲームを開始します。
「環境設定」からマスを10×10or20×20を選択できます。
「コマのデザインを変更」からデザインを球、立方体、正四面体から選択できます。

タブメニューの「その他」から、
「ハイスコア」でハイスコアを確認できます。

4、攻略

慣れるまでは一回のクリックで多く消すことではなく、オールクリアを目指してゲームを進めるほうがハイスコアを狙いやすいでしょう。

5、制作の上での工夫、要点

このゲームは Microsoft が提供している Visual C++ 2008（通称 VC）の WindowsForm アプリケーションで作りました。これはインターネットから無料でダウンロードして使うことができます。

このゲームは自分が教科書等を参考にせずに自力で作りました（数研は毎年恒例？の新しいパソコン作りで一年間の部費を使い切ってしまい、なかなか本が買えず、Windows フォームに関する本が殆ど無いのため ~~部費が少ない！~~）。
なのでいろいろと工夫をこらしました。ここではその工夫やこのプログラムの要点をいくつか紹介します。

Ⅰ）盤を作る

オセロでも何でもボードゲーム（SameMutch は違う？）のプログラムには、考える上でもボード（盤）を作ると良いと思います。盤は array()クラスを使用します。このプログラムでは

```
ban = gcnew array< Int32,2 >(42,42);
```

としています。当初は最大 40 マス×40 マスまで盤を変えられるようにしていたのですが、一つ一つのコマの大きさがとても小さくなって見にくいので断念しました。
こうすることで盤上の全てのコマの座標は ban(y,x)で表すことが出来るようになりました。この samemutch には緑、青、赤の三色が登場します。なので緑を 1、青を 2、赤を 3、何もない空欄を 4 とします。
つまり ban(y,x)＝3 のときはそこに赤色のコマの画像を表示すればいいわけです。このプログラムでは samemutch.exe がクリックされて samemutch のウィンドウが開くと同時に、

```
for(int x=0; x<Mx+2; x++){
    for(int y=0; y<Mx+2; y++){
        ban[y,x] = ran->Next(1,4);
    }
}

for(int x=0; x<1; x++){
    for(int y=0; y<Mx+2; y++){
        ban[y,x]=0;
    }
}

for(int y=0; y<Mx+2; y++){
    ban[y,Mx+1]=0;
}

for(int x=1; x<Mx+1; x++){
    for(int y=0; y<1; y++){
        ban[y,x]=0;
    }
}

for(int x=1; x<Mx+1; x++){
    for(int y=Mx+1; y<Mx+2; y++){
        ban[y,x]=0;
    }
}
```

＜プログラム①＞

これが実行されて盤が構成されます。
ここでは for 文が中心です。
for 文に出てくる int とは整数を定義するもので、int x で整数の x を定義したことになります。プログラムの世界ではモノを文字に置く時にはその文字をいちいち定義しないと使えないのです。

Mx とは盤の縦 横のマスの数です。10 マス×10 マスなら Mx＝10 となります。

ran とは乱数のことで、ran->Next(1,4)で指定された場所に 1 から 3 までの整数をランダムに入れなさい、というプログラムです。
x++というのは x=x+1 の略です。

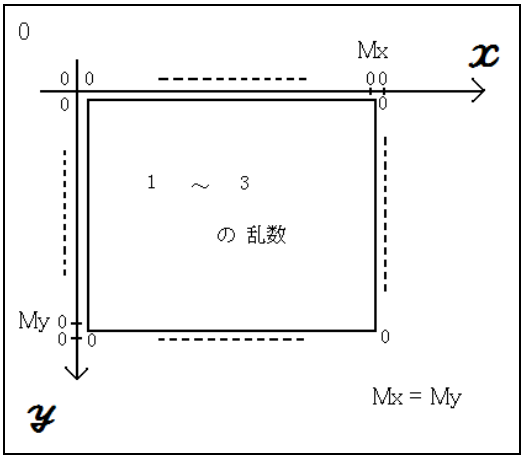
For 文とは決められた範囲において同じ作業を繰り返すループを行うときに使う文法事項です。一般的には

for(初期化；ループの継続条件；カウンタ変数の更新){
**　作業の内容**
}

という構文です。
ですから

```
for(int x=0; x<Mx+2; x++){
    for(int y=0; y<Mx+2; y++){
        ban[y,x] = ran->Next(1,4);
    }
}
```

は x が 0～Mx+1、y が 0～Mx+1 の範囲で ban[y,x]に 1～3 の乱数を入れなさい、というプログラムになります。



以上よりプログラム①によって盤は左図のようになるようにプログラムされることになります。

Ⅱ)コマが落ちてくるプログラム

このゲームでは消えたコマの上にあったコマは下に落ちてきます。そのプログラムは以下のようになっています。

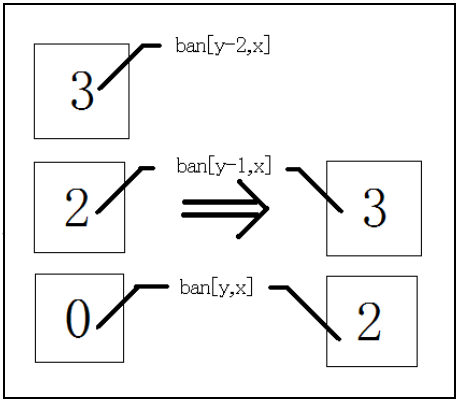
```
for(int x=1; x<Mx+1; x++){
    for(int y=1; y<Mx+1; y++){
        if(ban[y,x]==0){
            for(int i=y; i>0; i--){
                ban[i,x]=ban[i-1,x];
            }
        }else{
            ;
        }
    }
}
```

＜プログラム③＞

ここでは新しく if 文が登場します。If 文は一般的に

```
if(  条件式  ){
    実行する内容1
}else{
    実行する内容2
}
```

という構文です。
ですから x が 1～Mx、y が 1～Mx の範囲で ban[y,x]=0 のときには同じ列のそのコマより上のコマを一つずつ下にずらし、ban[y,x]=0 でない時は何もしないというプログラムになります(右図)。



ban[1,x]のように、表示されるコマの一番上の列が
ban[y,x]=ban[y-1,x]
という作業をすることがありますので、＜プログラム②＞のように**表示されるコマの周りを ban[y,x]=0 と決めておかないとエラーが起こります。**

Ⅲ）落ちるコマを判定するプログラム

このあとの「滑らかに落ちてくるプログラム」で必要になります。
盤wそ作ったときのように array()クラスを使用して

```
bandown = gcnew array< Int32,2>(42,42);
```

で bandown を作ります。そして

```
for(int x=Mx; x>0; x--){
    for(int y=Mx; y>0; y--){
        bandown[y,x]=0;
    }
}

int  maxdown=0;

for(int  x=Mx; x>0; x--){
    for(int  y=Mx; y>0; y--){
        if(ban[y,x]==0){
            for(int i=y-1; i>0; i--){
                bandown[i,x]=bandown[i,x]+1;
                maxdown=Math::Max(maxdown,bandown[i,x]);
           }
        }
    }
}
```

＜プログラム④＞

これで表示するコマそれぞれの落ちるコマ分を調べることができました。

Ⅳ）滑らかに落ちてくるプログラム

プログラム④で bandown を作ったのはこのプログラムのためで、プログラム④の後にこう書きます。

```
for(int k=1; k<=maxdown*4; k++){
    BanDraw2(k);
}
```

つまり BanDraw2()という関数の k に1以上 maxdown の四倍以下の k を代入します。その BanDraw2()関数は、

```
for(int i=Mx; i>0; i--){

for(int j=Mx; j>0; j--){

    if(ban[i,j]==0){

      g1->DrawImage(def,j*400/Mx-400/Mx,i*400/Mx-400/Mx);

    }else if(ban[i,j]==1){
      //前のコマを消す
      g1->DrawImage(def,j*400/Mx-400/Mx,i*400/Mx-
      400/Mx+Math::Min(400/Mx/4*k-400/Mx/4,bandown[i,j]*400/Mx));
      //  400/Mx/4 だけ下がったところにコマを描く
      g1->DrawImage(green,j*400/Mx-400/Mx,i*400/Mx-
      400/Mx+Math::Min(400/Mx/4*k,bandown[i,j]*400/Mx));//緑

    }else if(ban[i,j]==2){

      g1->DrawImage(def,j*400/Mx-400/Mx,i*400/Mx-
      400/Mx+Math::Min(400/Mx/4*k-400/Mx/4,bandown[i,j]*400/Mx));

      g1->DrawImage(blue,j*400/Mx-400/Mx,i*400/Mx-
      400/Mx+Math::Min(400/Mx/4*k,bandown[i,j]*400/Mx));//青
```

```
    }else if(ban[i,j]==3){

        g1->DrawImage(def,j*400/Mx-400/Mx,i*400/Mx-
        400/Mx+Math::Min(400/Mx/4*k400/Mx/4,bandown[i,j]*400/Mx));

        g1->DrawImage(red,j*400/Mx-400/Mx,i*400/Mx-
        400/Mx+Math::Min(400/Mx/4*k,bandown[i,j]*400/Mx));// 赤

    }
}
}
```

これでコマが1コマの 1/4 ずつ落ちていき、滑らかに落ちていくように見えます。

6)最後に

最後まで長々と書かれた退屈な文章をお読みいただき有難うございました。プログラムの知識が全くない人でも分かるように書いたつもりです。このプログラムは今現在自分が持っているプログラムの知識を最大限に詰め込んだもです。不備な部分もあると思います。楽しんで貰えたら幸いです。

コマンドプロンプトについて
（WindowsXP）

2009/11/3(Tue)　劉　嘉叡

パソコンを買ってから一度も使われたことのない方が大半だと思われますが。今回は使えると便利なコマンドプロンプトを紹介します。普通の操作ではできないこともコマンドプロンプトを使用すれば簡単にできることもあります。自分が実際に使ってみて役に立ったものを厳選して紹介します。

始めに　Windows には大きく分けて2系統があります。

95/98 系・・・Windows95・98・98SE・ME

NT 系・・・・・WindowsNT・2000・XP・Server2003

95、98系の Windows 上で MS－DOS のコマンド及びアプリケーションが動作するプログラム 及び仕組みを「MS-DOS プロンプト」と言います。NT系の Windows 上で MS－DOS のコマンド及びアプリケーションが動作するプログラム 及び仕組みを「コマンドプロンプト」と言います。

MS-DOS（Microsoft　Disk　Operation　System）とは、マウスを使わずもっぱら キーボードから命令（コマンド）を打ち込んで、アプリケーションを起動したり、ファイル 操作やシステム管理などの各種操作をする、文字ベースの操作環境 （CUI：Character　User　Interface）をもった OS のことです。

・MS-DOS アプリケーションが使える

・ワイルドカードを使って複数のファイル操作が一度で出来る

・繰り返す操作はバッチファイルで自動化できる

・Windows の細かいシステム設定や、ネットワークの調整ができる

コマンドプロンプトは MS-DOS のコマンドや MS-DOS 対応のアプリケーションが使えます。ただ、特殊なグラフィックスを使ったり、ハードウェアにアクセスして直接制御す るようなアプリケーションは動作しない事もあります。

コマンドプロンプトでは、OS のシステム管理やネットワークを取り扱うコマンドが 9x 系の Windows よりも充実しています。LAN 接続されたコンピュータの設定の確認や、接続不良を起こした際の設定の変更、他のコンピュータを遠隔操作するコマンドなども簡単に出来ます。システムに重大な障害が発生した際の、回復コンソールではコマンドプロンプトと 同様のコマンド実行環境で回復コンソール用のコマンドを使ってシステムの回復を行います。

1. コマンドプロンプトの起動と終了

まず、「スタートメニュー」→「すべてのプログラム」→「アクセサリ」を選び、その中の「コマンドプロンプト」を選びます。
続いて、終了の方法ですが二通りあります。
①通常使っているアプリケーションと同様に右上の「×」ボタンを押す。
②コマンドプロンプトに「exit」というコマンドを打ち込み「Enter」を押す。
実際に使用する場合は①がお勧めです。万が一間違ったプログラムを起動してしまったときは作業が終了するまで「exit」コマンドが打てなくなるからです。

2.ファイルシステムについて

ファイルシステムとは、「ディスクのどの場所に、どういったファイルやディレクトリ（フォルダ）が保管されているか?」という情報を管理するものです。WindowsXP 現在、使用されているファイルシステムは「FAT16」「FAT32」「NTFS」の 3 種類があります。FAT16 は MS-DOS から Windows95 までの OS で主に使われていたファイルシステムで、すべての WindowsOS で使用することができます。ただし、1 つのドライブとして管理できるハードディスクの容量は 2GB です。これに対して FAT32 は Windows98,98SE,Me などで主に使用されているファイルシステムで、管理できるハードディスクの容量は 2048GB になりました。FAT32 は FAT16 に比べてディスク管理の効率が良くなり、ディスクを有効に使うことが出来ます。NTFS は WindowsNT から登場したファイルシステムで、WindowsNT,2000,XP で使用できます。FAT16 や FAT32 と比較して、例えばファイルに障害が起きた場合にもファイルの構造を復旧することが出来るなど、**信頼性とセキュリティが高い**のが特徴です。また、個々のファイル情報についても FAT16 や FAT32 より詳細な情報を管理するインデックス機能が追加されました。これにより、ファイルやディレクトリ(フォルダ)の検索スピードがあがりました。

3.ハードディスクのデータ領域

ディスクは磁気を帯びた金属製の円盤状のものです。このディスクを始めてフォーマットする場合にディスクをいくつかの領域に分ける物理的フォーマットという作業が行われます。まず、木の年輪のように同心円状に区切ります。この年輪状の一つ一つがトラックです。それをさらに放射線状に一定の角度で扇形に区切ります。これがセクタです。ひとつのトラック内の複

数のセクタが合体してアロケーションユニット(クラスタとも言う)を構成します。このアロケーションユニットがデータを記憶する場合の最小単位になっています。コマンドプロンプトでディスクをフォーマットする場合オプションでこれらを区切る数値を指定できます。しかし通常は指定しなくても規定の値で問題ありません。

4. ディスクの検査を行う

WindowsVista ではコマンドプロンプトを使わなくてもできます。

chkdsk ［検査したいドライブ名］［コマンドオプション］

主なコマンドオプションの種類:

/f

ディスクの保存領域の破損などディスクのエラーを修復します。

/v

このオプションはファイルシステムによって動作が異なります。
FAT・FAT32 の場合、ディスクの完全なパスと名前を表示します。
NTFS の場合、検査の際にチェックしたファイルの一覧があればそれも表示します。

/r

不良セクタを見つけ修復できるデータ情報を回復します。

5. ファイルシステムを FAT から NTFS に変換する。

convert ［ドライブ名］［コマンドオプション］

主なコマンドオプションの種類

/fs:ntfs

デバイスのファイルシステムを NTFS に変換することを指定します。
このオプションは必ず指定する必要があります。

/v

コマンド実行の進行状況が%表記で表示されるようになります。

コンピュータ　チェスの歴史

和訳：前半　齊藤亘　後半　劉嘉叡

これは数学研究部の本棚にあった英語で書かれたチェスプログラムの本の和訳です。

コンピュータ　チェスの歴史を少し知るだけでコンピュータ　チェスが魅力的なものになるでしょう。

「The Turk」

最も古く有名なチェスマシンは「The Turk（トルコ人）」と呼ばれるものでした。それは1770年ごろにヴォルフガング・フォン・ケンペレンによって作られました。18世紀の間、それはヨーロッパ中の宮廷の王室に置かれていました。チェスプレイヤーとしてよりも将軍として有名だったナポレオンを打ち負かしました。

「The Turk」は大きな棚と、トルコ人の格好をしたマネキンからできています。左手で駒を動かし、すばらしいチェスをしました。大きな棚にはハンドルや扉があり、その扉を開けると滑車やケーブルがありました。すばらしい動きができる理由は、中にプロのチェスプレイヤーが隠れているからでした。疑い深い人は、機械がチェスをすることなんて出来ないと疑いましたが、この種のチェスマシーンは19世紀の末まで人々を驚かせ続けました。「The Turk」は1854年に火事によって無くなりましたが、設計図によって再建することができました。そのチェスマシンは中に人間が入って動作させていたといういたずらでありましたが、人々はマシンが何世紀にもわたってチェスをしているかのような夢を見ました。

最初の本当のチェスマシンは1890年にスペイン人の科学者、レオナルド・トーレス・Y・ケベードによって作られました。それはルークやキングを使ってチェックメイトすることが出来ました。とても単純な勝ち方しかできませんでした。

「The Torres Machine」

レオナルド・トーレス・Y・ケベードはオートメーションに関する学説を論証するために機械を作りました。そのチェスマシンはワイヤーやスイッチの複雑な仕組みによって出来ていて（これはトランジスターが開発されるはるか

昔のことです)、当時の最も精巧な機械の一つでした。またそれはワイヤーやスイッチからできた最も精巧なコンピュータの一つでした。それは現在も動き、マドリッドの科学技術専門学校の博物館で見ることが出来ます。

1949年ベル研究所所属のクロード・シャノンは「Programming a Conmputer for Playing Chess」という題名の論文を書きました。その論文ではチェスのプログラムの書き方を説明していました。それは、コンピュータがまだきわめて原始的である初期の頃で、彼の論文は「知識が必要な仕事をコンピュータにさせるようにプログラムすることができる」ということを人々に納得させるものでした。この論文の驚くべきことは今日でも、ほとんどのチェスプログラムは、この論文をベースに作られていることです。

最初に実際のチェスプログラムは1951年にアラン・チューニングによって作られました。彼のアルゴリズムは複雑すぎて当時はプログラムに利用することが出来ませんでした、なので彼は手作業で計算しなければなりませんでした。

最初のコンピューターで動くチェスプログラムは1956年に作られた「ロス・アラモス・プログラム」でした。当時のコンピュータはあまり容量が大きくなかったので、チェスの盤は6×6で、ビショップの駒を取り除いていました。(いく人かの著者はソビエトが少しだけ早く開発していたと著しているが、あまりよく分からない。)そのプログラムは毎秒11000の命令の速さの「マニアックコンピュータ」で動きます(今日ではIBMのコンピュータは毎秒500000の命令です)。またこれは初めて人を打ち負かしました。

最初の完全なチェスプログラムはおそらく1958年にアレックス・バーンスタインに作られたものでした。その後、数年間さまざまなプログラムが作られ、コンピュータはどんどん速くチェスができるようになりました。1966年にはアメリカとソビエトのプログラムの対決が準備されました。ソビエトのチェスプログラムはアメリカのチェスプログラムを打ち負かしました。ロシア人は昔からずっとチェスが得意です。次の年、実際のチェストーナメントで始めて人間を打ち負かしたのは、リチャード・グリーンブラットが作った「マックハック」でした。1970年、初めて4つのプログラムによるチェストーナメントのACM国際大学対抗プログラミングコンテストが開かれました。優勝は、デービッド・スレート、ラリー・アトキンスらが作った「Chess 3.0」でした。ACM国際大学対抗プログラミングコンテストはそれ以来毎年開かれており、今日最も重要な大型コンピュータのチェストーナメントとなりました。

1960・70年代は、ばかばかしく違法なコンピュータ崩壊の時代であり、そしてプログラマーが極度の楽天主義になった時代でした。1968年、プロチェスプレイヤーであるデービッド・レビーに対して四人のコンピュータ専門家が3,000ﾄﾞﾙをめぐって賭けをしました。その賭けというのは彼がこれから10年以内に、最高のチェスプログラムには勝つことができないというものでした。レビーは1978年に「Chess 3.0」の後継である「Chess 4.7」を打ち負かし、賭けに勝ちました。もし初期のコンピュータチェスの時代のことを知りたければ、コンピュータ サイエンス プレス社出版のデービッド・レビー著の「Chess and Computer」を読むと良いでしょう。

1970年代に作られた最良のチェスプログラムは「Chess 4.N」でした。それは今でも最良のプログラムの一つですが、現在では「Nuchess」と名前を変えました。この本が紹介しているチェスのプログラムを含む、現在の最高のプログラムは、「Nuchess」を基礎に作られています。チェスプログラムのチェスの強さはプログラムが良くなったというよりも、コンピュータの性能が良くなったことによります。

1980年、ベル研究所のケン・トンプソンは、カスタマイズされたベルシステムを作り、チェスの大会で優勝したしました。（トンプソンはUNIXオペレーションシステムを開発したため、彼はある程度ベル研究所で自由に行動できました）。ベルは130ポンドの箱で、コンピュータとチェス専用のハードウェアが入っています。それは特別なハードウェアなので、1秒間で10万ものチェスの局面を分析することができます。ベルは国が国家の安全を脅かす程凄いとみなした最初のチェスコンピュータになりました。トンプソンはそれをモスクワのチェスの展示会に持っていこうとしましたが、アメリカの権力者は、それがソビエト軍にとって欠かすことのできない軍事上の重要なものに成り得ると信じていたため、空港で押収されました。

今日での最良のチェスプログラムはベルかクレイの作ったクレイ・ブリッツ、ロバート・ヒラットでしょう。クレイ・ブリッツは特別な実験的なシステムだから、動かすには当時の最速の16プロセッサーのクレイコンピュータが必要でした。パソコンの強さは2200ELOぐらいでしょう。だからと言って皆がベル研究所のクレイにアクセスしたわけではなかったのです。彼は70年代終わりにパーソナルコンピュータで動くチェスプログラムの開発に着手しました。

1978年、初の商業用のチェス専用のマイクロコンピュータである「チェスチャレンジャー10」がアメリカで製作されました。それはそんなにうまくはプ

レイできなかったが、まずまず成功し生産が始まりました。同じく、香港にある2,3の会社が、チェスのコンピュータの生産を開始しました。今日の多くのチェス専用のコンピュータは香港で作られています。しかし、最も良いプログラムはまだカリフォルニアで製作されています。マイクロコンピュータで動くチェスはプログラムの改良のおかげで劇的に強くなっていきました。ほとんどの商業用のチェスはまだ6502チップを使用しています。(「Apple Ⅱ」「Commondore」「Atari」コンピュータにも使用されている)新しい16や32ビットのチップは、普通のよりは速いがチェスのプログラムには必要ありません。チェスのプログラムのためだけなら、4MHz6502と8MHz68000はほぼ同じ速さです。初期のころの、マイクロコンピュータのプログラムは4MHzのプロセッサーを使用して1秒に800の局面を分析しました。これと比べてクレイ・ブリッツの「Cray-1」のコンピュータは、1秒間に15000もの局面を分析し、前者の20倍です。チェス・パス・プログラムの場合は、5MHzのIBMのPCを使っているときで1秒間に50の局面を分析します。

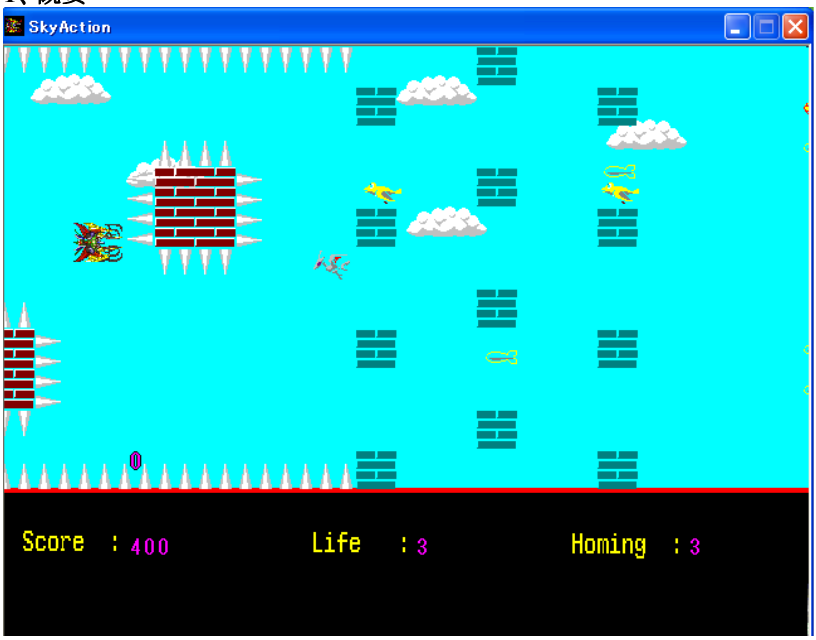
もしこれらのプログラムや作った人に会いたければ、毎年開催されているチェスのトーナメントに行きましょう。大型のコンピュータのためのACMトーナメントは、毎年アメリカでACM競技連盟によって開催されています。マイクロコンピュータのためのマイクロコンピュータワールドチャンピオンシップは毎年，普通はヨーロッパで開催されます。トーナメントは古顔の会う場所、お互いに考えを交換する場所、コンピュータが難しい計算をしているときに少しのビールを飲み、楽しい時間を過ごす場所となります。何とかして、コンピュータのチェスは、本物のチェスよりも面白いものとなりました。大抵のプロのチェスのプログラマーはとてもチェスが強いです。しかし実際、彼らのほとんどが自分の作ったプログラムを打ち破ることができません！

-- この本は1980年台の出版のため、ここに書かれているデータは決して最新のものではありません。1996年ごろにはコンピュータは人間のチャンピオンレベルに達していて、2007年には人間に負けないレベルになっています。

SkyAction

59Ｋ　2Ｃ27　新田　泰大

1、概要



今回の作品は横スクロールシューティングです。
C++の DX ライブラリを使用して製作しました。
ステージは 1 から 6 まであり、様々な特徴があります。

2、遊び方

①操作方法

★メニューでの操作

Z キー　ゲーム本編へ

X キー　操作方法の表示

★ステージセレクト

十字キーで選択して、スペースで決定します。

★ゲーム本編

Zキー　まっすぐ飛ぶ弾を撃つ。
Xキー　敵に絶対あたる弾を撃つ。(ボス戦では不可
Cキー　斜め下方向の弾を撃つ。
・・・・・その他裏技あるかも。自分で探してみてください。

②ルール説明

・基本的にすべての敵を倒していけばOKです。
・敵キャラは3種類いてそれぞれ違う動きをします。
・ボス戦までに、Homing(追尾弾)をためとくとその3倍の数だけライフが増えます。
・Homing(追尾弾)は1つにつき80発までしか発射できません。残りの弾の量は表示されません。
・ステージ1はボスがいません。

③改造方法について

プログラムに関しては、C/ProgramFiles/skyaction にあります。
(↑デフォルトのインストール先)

yokogame01.cpp・・・・・・・ゲームの骨組み
jichara.cpp・・・・・・・・・・・・自キャラ関係・あたり判定
tekichara04.cpp・・・・・・・・ボス戦
map.cpp・・・・・・・・・・・・・・マップの読み込み
tekitama.cpp・・・・・・・・・・・ザコ敵・ボスの弾発射関係
spgame.h・・・・・・・・・・・・・グローバル変数など

Resource.h は無視してください。

改造は基本的に自由ですが、変数名など汚く、分かりにくいものがたくさんあると思います・・・

④感想

今年は夏休みの活動時間が短いため、たいしたものができなかった。今回はこれを反省したい。

来年は3Dゲーム(DirectX9.0を使用)をつくってみたい。

SkyAction プログラム説明

59K 2C27 新田 泰大

今回のゲームをつくるのに一番苦労した点は、**敵追尾弾(ほぼ無敵)プログラム**(本来こんなのは認められない。**シューティングとしてズルい**)を入れてみた。まあそんなことはいいとして、数学に出てくる三平方の定理をゲームプログラミングにもつかえるということを紹介したいと思う。

基本事項

※C言語では右図のように数学とはちがって左上が(0,0)である。
→図1参照

※a=bというのは、数学でいうと a と b が等しいことをあらわしますが、C言語では、**b という変数 a に入ってる数を a に代入する**ことを表します。つまり、x=x+5 これはxにx+5を代入する、つまりxが2のときこれで7になります。

※掛け算は×ではなく*であらわします。割り算は÷ではなく / です。
例) 4=2*2 3=6/2

※三平方の定理とは次のことを言う。→図2参照

※C言語では特殊な場合を除いて、{や(以外は;(セミコロン)が必要です。

図1

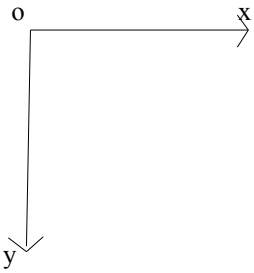
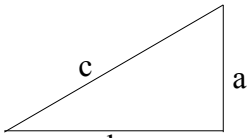


図2



$$c^2 = a^2 + b^2$$

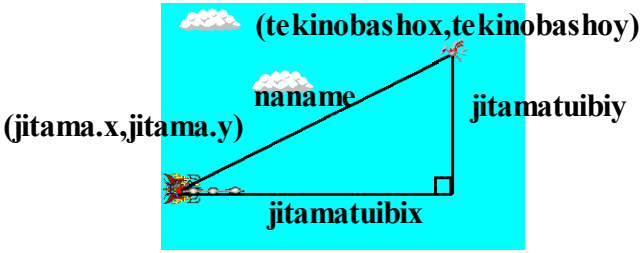
変数名の紹介

jitama.x, jitama.y (じたまx,y) ・・・・自分の弾のx座標、y座標です
tekinobashox,tekinobashoy(てきのばしょx,y)・・・敵のx座標、y座標です
naname(ななめ)・・・・・・・・・・・・・・・・・・・・・・斜めです。詳しくはあとで
jikitamatuibix, jikitamatuibiy(じたまついびx,y)・・・これもあとで
Jitama.yobi1,jitama.yobi2(じたまよび1、2)・・自分の弾のx座標・y座標
の移動量です

プログラム(①～⑧は分かりやすくするためにつけた番号です。
プログラムではありません)

```
① jikitamatuibix = tekinobashox-jitama.x;
②jikitamatuibiy = tekinobashoy-jitama.y;
③naname = sqrt(jikitamatuibix*jikitamatuibix
                                    +jikitamatuibiy*jikitamatuibiy);
④jitama.yobi1 = jikitamatuibix/naname;
⑤jitama.yobi2 = jikitamatuibiy/naname;
⑥ jitama.x = jitama.x + jitama.yobi1;
⑦jitama.y = jitama.y + jitama.yobi2;
⑧DrawGraph(jitama.x,jitama.y, tama, TRUE);
```

プログラム解説



①ｘ座標の長さを求める

自機の弾と敵の弾の座標の差を取ります。それを、jikitamatuibix に代入します。

② y座標の長さを求める

① と同じことを y 座標に対しても行います。

③ 斜めの長さを求める

sqrt とは数学でいう√のことです。つまり sqrt(16)なら結果が 4 になります。そこで三平方の定理を使うと、naname（斜め）の長さがでるのです。
もちろん、jitamatuibix などが負の数であっても、naname は正の数になりますね。

④x座標の差を斜めの長さで割る

これは、斜めの移動量は 1 にすることを前提にしています。
例えば、斜めの長さが10で x 座標の差が5なら斜め方向に1だけとばすので5÷10=0.5だけ弾を x 座標方向に移動すればよいでしょう。
もちろん、jitamatuibix が負の数の場合、naname が正ですので、代入する jitama.yobi1 には負の数が入り、左方向に動きます。
（もちろん斜めに飛んでいるように見せるには⑤をして、x 座標と y 座標を両方移動させればいいのです。）

⑤y 座標の差を斜めの長さで割る

④ と同じことを y 座標に対しても行います。

⑥弾の x 座標の移動量を弾の x 座標に足す

ここで初めて自機の弾の x 座標が変化します。

⑦弾の y 座標の移動量を弾の y 座標に足す

⑥と同じことを自機の弾の y 座標に対しても行います。

⑧指定された座標に画像（自機の弾）を表示します。

変化した(jitama.x,jitama.y)に弾が表示されます。

①～⑧のことを繰り返すことで自機の弾が敵にあたるのです。

感想

今回は、数学の公式がプログラムで使えるということを解説しました。
ほかにも、三角関数や加法定理など僕自身よくわからない定理が今回のプログラムにもあります。
これを読んでプログラムと数学が関係あることを知ってほしい。
最後まで読んでいただいて有り難うございました。