ソフトゼミ B　第 6 回

プレイヤーの移動と重力

シューティングゲームの制作お疲れ様でした！今回からはアクションゲームのサンプルを制作していきます。アクションゲームとシューティングゲームの違いは、まず重力があるというのが上げられます。これから 3 回に渡って行われるアクションゲームでは「プレイヤーの移動と重力」「ブロック判定」「スクロール」の 3 つを学習していくつもりです。

◆ 準備

今回はアクションゲームの初回ということで、第 1 回のように新しくプロジェクトを作るところから始めましょう。ここまでは復習なのでゼミ B 第 1 回の資料を参考に、プロジェクトの新規作成から main.cpp の作成､DxLib の設定を行なってください｡また付属の img フォルダと map フォルダを制作中のプロジェクトフォルダの中に入れてください。

◆ プログラムの解説

設定を行ったら、付属のソースコードを main.cpp にコピペしてください。
しかしながらこの状態ではまだ動きませんので、以下の解説を参考にプログラム中の空欄を穴埋めしていってください。

◆ move 関数の穴埋め

まず、move 関数とは、重力速度、移動速度、主人公の向き、設置判定の処理を行っています。

重力加速の処理は、player.jfly は 0 の時に着地しているので、0 以外の時に飛んでいる状態です。このときに、下向き（Y 軸方向）の速度を GRA だけ増やします。しかし速度が MAXSPEEDY 以上の時は、これ以上重力加速をしないようにします。

これ以降では穴埋めをしてもらいます。ヒントは図の下にあるので分からない場合は、参考にしてください。

```
GetHitKeyStateAll(keyState);                    //キーボードの状態を取得
        if(keyState[【穴埋め 1】]){
                if(player.kabejl==0&&player.x>0){   //プレイヤーの左向き操作
                        player.x-=4;
                }
                player.dire=1;                  //左向いてます
        }
        else if(keyState[KEY_INPUT_RIGHT]){
                if(player.x <= WINDOW_WIDTH-HERO_SIZE ){
                        //プレイヤーの右向き操作
                        【穴埋め 2】;
                }
                player.dire=0;                  //右向いてます
        }
```

【穴埋め 1】キーボードの右矢印が押されたときの判定です。
【穴埋め 2】プレイヤーの x 座標を右に 4 づつ進めます。

```
if(player.jfly==0&& keyState[ KEY_INPUT_X ]){               //ジャンプ
                【穴埋め 1】=1;              //飛んでます
                player.vy=-15;
        }
```

ジャンプ判定を行います。X キーが押されている＆player.jfly は 0 の時、つまり着地している時にジャンプするようにします。処理としては player.jfly を 1 にして空中にいる判定にした後で、プレイヤーの Y 軸速度を-15 にすることで、上方向へジャンプします。
【穴埋め 1】は着地判定を変えています。

```
        【穴埋め 1】;                  //x 軸方向移動します
        player.y+=player.vy;        //y 軸方向移動します

        if(【穴埋め 2】{     //プレイヤーの y 座標が画面サイズ-ブロック一マスより大きいとき
                player.jfly=0;                  //着地状態にする
                player.vy=0;                    //落下速度を 0 にする
```

```
                【穴埋め 3】
        }
        else{
                player.jfly=1;                  //それ以外の時は空にいるよ！
        }～～～～～～～～～～～～～～～～～～～～～～～～後略～～～～～～～～～～～～～～～～～～～～
```

【穴埋め 1】最終的に上で変えたプレイヤーの x 方向の速度を足します。
【穴埋め 2】プレイヤーの y 座標が画面サイズ-ブロック一マスより大きいときの判定。
【穴埋め 3】プレイヤーが地面にめり込んでしまったとき、地面の上に戻します。

◆ draw 関数の穴埋め

draw 関数では、主に背景の表示と主人公の表示、敵の表示、デバッグ用の数字を出力しています。

今回はアクションゲーム初回ですので、最低限の描画しかしません。ですので単純な今回で描画の仕組みを理解するようにしてください。

```
void draw(void){
        ClearDrawScreen();                  //画面クリア
        for(int i=0;i<BLOCK;i++){
                DrawGraph(i*BLOCK,0,img.haikei,TRUE);
        }                       //背景かく
        for(int j=0;j<【穴埋め 1】;j++){                    //y 軸 10 分割単位で
        for(int i = 0 ; i < WINDOW_WIDTH/BLOCK; i++){//x 軸 32 分割いろいろで
                if(i<WIDTHSIZE&&i>=0){   //横の部分より狭い範囲でマップ判断
～～～～～～～～～～～～～～～～～～～～～～～～中略～～～～～～～～～～～～～～～～～～～～
        if(【穴埋め 2】  == 0){                    //主人公書くよ
                DrawGraph(player.x,player.y,img.migi,TRUE);       //右左その辺
        }
        else{
                DrawGraph(player.x,player.y,img.hidari,TRUE);
        }
```

ここで背景の描画を行います。

【穴埋め 1】 では背景の y 座標は 0 からどこまでなのかを考えてみてください。
【穴埋め 2】は、主人公の向きの変数はなんでしょう。
さい。

◆ title 関数の穴埋め

title 関数では、主にスタート画面の表示と X ボタンを押さない限り無限ループをするということをしています。

```
void title(void){
        ClearDrawScreen();
        DrawGraph( 0 , 0 , 【穴埋め 1】 , TRUE ) ; //タイトル画面を描画
        ScreenFlip();
        while( keyState[【穴埋め 2】 ] != 1 && keyState[ KEY_INPUT_X ] != 1){ //エスケープキー、X が押されると
                GetHitKeyStateAll(keyState);
                if( ProcessMessage() == -1 ){
                        break ; // エラーが発生したらループを抜ける
                }
        }
}
```

ここの【穴埋め 1】は、タイトル画面で表示される画像はなにか？（なんという名前で保存されているか？）というのを考えてみて、穴埋めをしてください。

【穴埋め 2】は ESCAPE キーが押されるときの判定。