グループ番号

K

グループ名

おかあさんといっしょ（砂糖　大　路）

メンバーの学籍番号と名前

　小下　元路　４１４８２８

　石原　俊　　４１４８０４

　山下　大志　４１４８５８

作品名

テトたん！

概要

　各教科の単位を取得するべく奮闘するテトリスです。最後まで生き残れればめでたく単位を取得できます。

ゲームを起動するとタイトル画面が表示され、ステージ開始前に職業と学科、アピールポイントを選択し、個性豊かなプレイヤーキャラで敵に立ち向かいましょう。

テトリミノを操作して課題をこなして行き、道中の敵（講義やレポート）を倒していきましょう。先生方の思いやりに答えてすべてのレポートやテストをこなしましょう（消してこなしましょう）

見事すべての試練を乗り越え、単位を取らせてあげましょう。

テトリス初心者も楽しめるように、スコアからのダメージ計算だけでなく（スコアからの算出だとテトリス連続が大きく有利になる）、消したラインの数からもダメージを与えられるようになっています（一列ずつ消していっても割とダメージがでる）

ちなみにスコア計算は物理、ライン計算は魔法です。

回復魔法はステータスに影響されません。

あと敵は大学生の勉強を妨げる誘惑や厳しいレポート、講義です

変更点

東方がテトリスになりました。

分担

石原　　俊：　テトリス基本部分

小下　元路：　テトリスの特殊部分、RPGっぽいところ、画像の加工、およびフレーム、ＵＩ画像、テトリミノ画像、テトリス時のボード作成等、その他もろもろ

山下　大志：　RPGっぽいところ

プログラムの使い方と解説

起動後はタイトル画面が表示されます。

ハイスコアを選ぶと今までのハイスコアが上位１０位まで表示されます。

ゲームエンドを選べばプログラムが終了します。

ゲームスタートを押すとキャラクター作成画面に入ります。

ジョブを選んで基本パラメータを決定し、アピールポイントで好みのパラメータに調整しましょう。

ジョブはそれぞれステータスに個性があるのでプレイスタイルに応じたものを選ぶと有利です（テトリスぶっぱは物理系、ちょこちょこ消すなら魔法系）

次に攻撃魔法を一つ選びます。

基本的に必要ラインが多いほど１ライン当たりのダメージは多くなります。

回復魔法も同様です。

また、特徴、魔法は選ばないことも可能なので自身のある方は試してみるといいかもしれません。職業も「１２年目」に関しては意図的に弱く調整してあります。

テトリスはｚで←、ｘで→回転です

↑を押すとクイックドロップができます。

敵は一定時間ごとに物理攻撃、一定数テトリミノを落とすと魔法攻撃を仕掛けてきます。

データ構造と作成した関数の説明

TIMER（テトリスの計算部分）

PAINT（テトリスの描写部分)

typedef struct{

//フラグ、見た目、出現するカウンタ、状態（カウンタいらないかも)

int flag,lookknd,encnt,state;

//最大HP,HP,攻撃力、防御力、魔力、魔法防御、命中力、スピード、攻撃間隔のフレーム、特技番号の配列、

int hp\_max,hp,atk,def,inte,res,hit,spd,atk\_num,spe[SPE\_MAX];

}enemy\_t;

typedef struct{

//最大HP、HP,最大MP,mp,攻撃力、防御力、魔力、魔法防御、命中力、スピード,特技、特徴

int hp\_max,hp,mp\_max,mp,atk,def,inte,res,hit,spd,spe[CH\_MAG\_MAX],chara[CHARA\_MAX];

int job\_name;//職業名

char name[64];//職業名のままにするかもしらん

//判別に使おうかと思って名前intとchar両方作ったけどいらんかったかもしらん

int mag;//０は未所持

}ch\_t;

//hit,spdは実装未定、特技も実装未定、特徴はつける予定

あと職業のデータ入れる構造体と特技入れるやつと作って

それぞれ関数ポインタで呼び出す感じにする

敵の出現順とかはcsvファイルに記述してそっから読ませる

ハイスコアもファイルから

キーコンフィグは実装未定

メインの冒頭でチームロゴ表示、表示している間に画像を読み込ませる

void timer\_handler(void){

switch(state){

case s\_title://タイトル

title\_top();　タイトル表示、プレスＺキーのとこ

break;

case s\_menu\_title://タイトルメニュー画面

title\_menu();　ゲームスタートエンド、ハイスコアの表示

break;

case s\_hi\_score:

hiscore();　ハイスコア描写用の関数

break;

case s\_ini\_select\_ch:

load\_job();　　ｃｓｖファイルからジョブの設定を読み込んでch\_t型の構造体に登録する

ch = job[0];　自キャラであるch\_t型のchに初期ジョブとして代入

ch\_menu\_select =1;　次画面の初期化

state = s\_select\_ch;

case s\_select\_ch://ジョブ選択とか

select\_ch();　ジョブの選択、特徴の選択、白、黒魔法の選択

break;

case s\_select\_stage://ステージ選択画面（実装未定）

strcpy(story\_name,"./csv/debug.csv");　ストーリーファイルの読み込む場所指定、拡張性を見込んで引数からファイルを指定するようにした

load\_story(story\_name);　ｃｓｖファイルからステージの情報を読み取って敵を登録する

state = s\_ini\_battle;

cnt =0;

ene\_atk\_cnt = cnt;　カウント初期化、このカウントは敵の攻撃に使う

break;

case s\_ini\_battle:

ini\_battle();

戦闘に入る前の初期化をいろいろ

特徴と選択した魔法を自キャラに登録、各変数の初期化等

state = s\_battle;

break;

case s\_battle://ゲーム本編

battle\_main();

テトリスと戦闘部分本体、中でだいぶ分岐してる

キャラ計算（物理、魔法、その結果の敵のＨＰチェック）

敵計算（物理、魔法、その結果のキャラのＨＰをチェック）

敵と味方とかの描写部分

みたいな構造

break;

case s\_battle\_clear:

EzDrawStringB(100,100,"ステージクリア!!");

EzDrawStringB(100,200,"また別の組み合わせでも遊んだりしてね！");

EzDrawStringB(100,300,"Please\_press\_z");

clear=1;

break;

case s\_game\_over:

EzDrawStringB(100,100,"ゲームオーバー");

EzDrawStringB(100,200,"Pleas\_press\_z");

clear=0;

break;

case error\_ene\_max:

EzDrawStringB(100,100,"敵登録数がオーバー");

break;

case error\_read\_job:

EzDrawStringB(100,100,"ジョブファイルがありません");

break;

case error\_job\_max:

EzDrawStringB(100,100,"ジョブ登録数がオーバー");

break;

case error\_read\_story:

EzDrawStringB(100,100,"ストーリーファイルがありません");

break;

case s\_check\_score:ゲームクリアないしゲームオーバーなったらここへ

show\_score();　最終スコアを計算して表示、ハイスコアランキングに入るなら登録

break;

}

// EzPutB(0,0,title\_pic);

debug();デバック用、今は死んでる

cnt++;　カウンタをマイフレーム増やしていく　６０ｆｐｓ

EzShowBuffer();

}

規定の４つの箇所

ポインタ

戦闘部分(stage.c)で自キャラの魔法を指定するときに関数ポインタを使用

文字列

パラメータ表示、スコア、ライン表示等

構造体

敵のパラメータ、自キャラのパラメータ、ジョブの設定を構造体で管理

ファイルの読み書き

ステージデータ、ジョブデータをｃｓｖから読み取り

ハイスコアを読み込み、更新時は書き込み

技術的に工夫した点

ループのなかにスイッチ文で分岐をさせることにより、複雑な処理を可能にするのと同時に拡張性高めた。

連続的に文字を表示する場面では、起点となる座標と行間の幅を指定してループ処理を行った。

タイマーハンドラーとキーハンドラーの中身はスイッチ文で連動しているので、場面ごとに欲しい操作を行える。

スイッチ文に列挙体を用いることで重要な部分のループを見やすくした。

バランス調整の際に変更の多そうなステージデータとジョブデータをｃｓｖファイルから読み込むようにすることで効率化をはかった。

芸術的に工夫した点

画像とＵＩはかなり力をいれましたです。

ＵＩ用の文字の画像、テトリミノ、テトリスの枠等結構自作しています。

ｘｐｍは透過が無いので事前に透過処理をした画像を用意することで疑似的に再現しています。

他にも画像の色数を２５６から６４まで現職させて軽量化を図る等、画像の編集と画像や数字の配置、操作性の良さにはかなり力を入れています。

また、ゲームとしてもきちんと成り立つようになるべくテトリス部分と戦闘部分を左右で分けてプレイヤーがテトリスに集中できるようにしています。

また、見ている人は戦闘部分もあるので通常のテトリスよりも飽きにくいかと思います。

利用したフリー素材

http://k-0n.com/

けいおん風ジェネレーター、タイトルロゴ

http://www.squarespace.com/logo

Squarespace Logo、グループロゴ

<http://etolier.webcrow.jp/index.html>

エトリエ、敵画像

<http://k-after.at.webry.info/200909/article_1.html>

きまぐれアフター、背景の森

他は自作