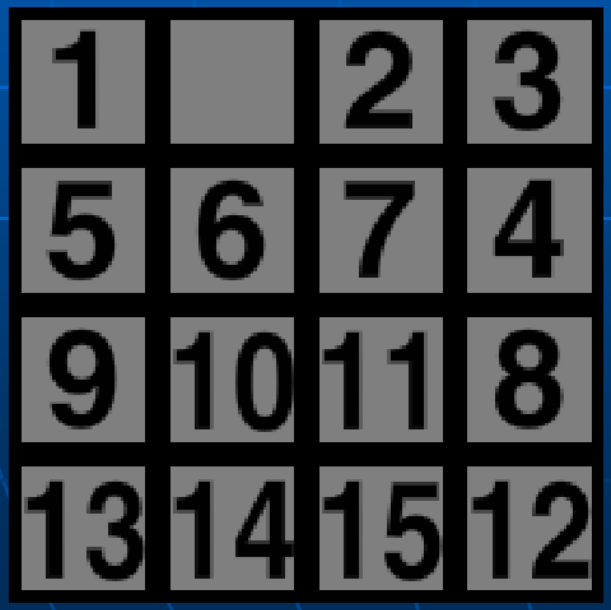
15パズルを作ってみた

CUI～GUI版まで

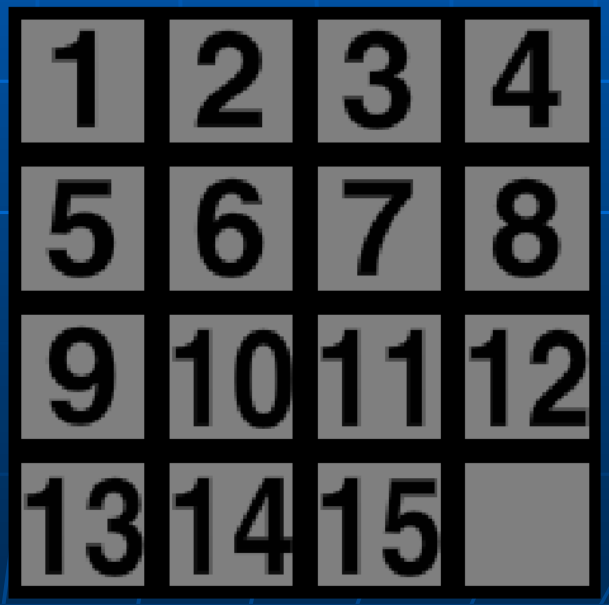
15パズルとは

- 4×4のマスにランダム並べられた1～15までの数字を順番に並び替えるパズル

クリア前



クリア後

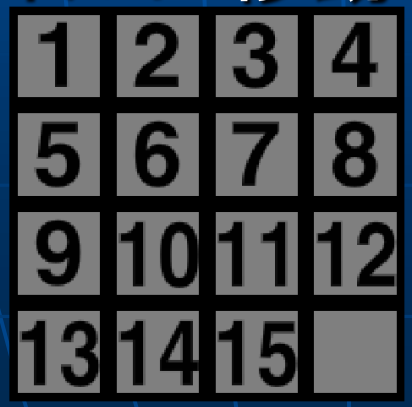


とりあえず遊んでみよう！

URL:https://scratch.mit.edu/projects/125032169/#player

これは僕がScratchで試作した15パズルです。

数字が書かれたタイルをクリックするとその数字の隣接しているところに空白のタイルがあればそのタイルに移動します。この作業を続けていって



←のような盤面にします。

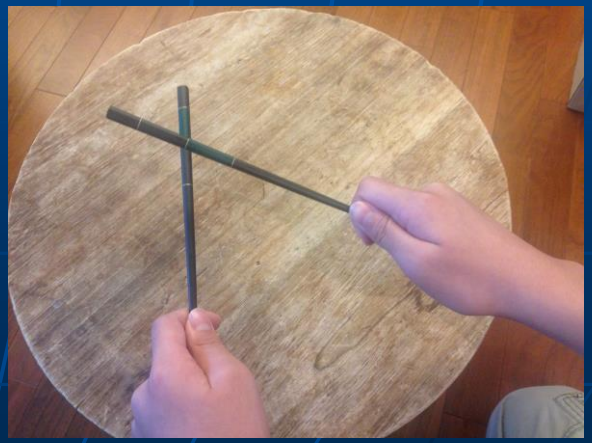
コンソール版15パズル

コンソール版の15パズルを作ってみました。
これは動かしたい数字を入力してスクラッチ版のようにクリアの状態まで移動させます。
開発環境：VisualStadio2015
開発言語：C++

グラフィック版15パズル

続いてグラフィック版です。
これはスクラッチと操作は同じです。
ちなみにこれはDxLibというライブラリを使用しています。
グラフィック版は後ほどお見せします。

開発環境：VisualStadio2015
開発言語:C++
絵素材提供：母
音提供（タイル移動）：箸(父の物)
音監督：父



DxLib（Dxライブラリ）について

DXライブラリとは、Windowsソフトの開発に使う、DirectXやWindows関連のプログラムを使い安くまとめた形で利用できるようにしたC＋＋言語用のゲーム開発ライブラリです。無料で使用可能。

主な機能として

URL:http://dxlib.o.oo7.jp/dxfunc.html

コンソール版からグラッフィク版へ

さきほどのコンソール版の数字タイルをグラフィックにすればいいことです。

あとはマウスで操作できるようにします。

じゃあコンソール版の中身は？

必要な関数↓（ページ1）

①初期化する関数(配列の値の初期化など)

②数字を表示する関数（データに基づき数字を表示する）

③入力受け取り（ユーザーから数値を受け取る）

④データを更新する関数（ ③で受け取った数値をデータ上で移動させる）

データ＝数字の状態を保管する配列

つづき・・・

必要な関数↓（ページ2）
⑤クリアしているか調べる関数
クリア後の配列の状態↓
 1 ， 2， 3， 4，
 5 ， 6， 7， 8，
 9 ,10, 11,12
13,14, 15,0（空白）
上のように配列が出来ていたらクリア！！

順番

今見てきた関数を組み合わせる順番は

初期化する

クリアするまで繰り返す{

数字を表示

↓

入力を受け取る

↓

更新をする

↓

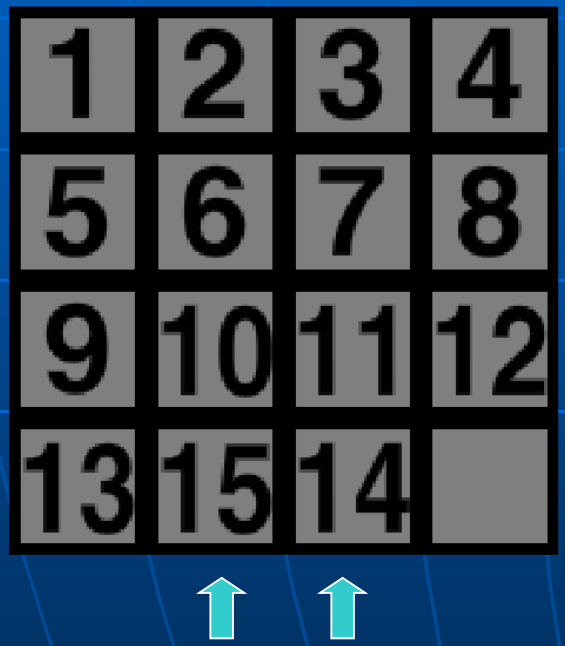
クリアしているか確かめる

ところで・・・

最初の盤面を生成するにはどうするか？
これは単純に初期データ(配列)を同じ数字が入らないようにランダムに0～15(0は空白)を並べていけばいいでしょ。。。
↑と、僕は最初に思いました、が。。。。。。。

ところが・・・

さっきの生成方法で生成したパズルをやったら
100手、150手でも解けないので調べてみたら
どうやら解けないパターンがあるようなのです。
例：



上のように、進めていって最後に15と14が入れ替わった
パターンになると、絶対にクリア不可能だそうです。
（Wikiペディア参照）

そしたら・・・

そしたら、どう生成したらいいんだろう？
と、いうことでたどり着いた結論がクリア後の状態から適当に数字をシャフルさせる関数を作る、というものです。

ではざっと中身の説明

- シャッフルの回数は自分で決められる（一回のシャッフルで必ずランダムな数字が動く）
- 数字の表示は繰り返しで表示する
- 入力で受け取る数値はint（整数）型
- 更新は↑で受けった数値を動けるかどうか調べて（一次元配列を二次元配列にみなして隣接しているところに空白があるか調べて）データ上で移動させる。

重要関数たち

今までの関数はこの重要関数を束ねたものです。
例：↓
関数MoveTile(int num)（更新（整数　入力された数値））
↑の中身は
int Tx, Ty,TMA;//Tx,Tyはタイル（数字）の配列上の座標、TMAは
TileMoveAngleの略でタイルの動ける角度0なら上、1なら右、2なら下、3なら左
GetTilePoint(Number, &Tx, &Ty);//入力された数値の配列上の座標をTxと
Tyに代入する見つけられなかったらTx,Tyには-1が代入される。ちゃんとポインタ
でやっています。
if (Tx == -1 || Ty == -1) return;//見つけられなかったら終了
TMA = GetTileMoveAngle(Tx, Ty);//タイルの配列上の座標からたいるの隣接
　していい
るところに空白があるか調べて（空白がない場合は-1が返る）動ける角度を返す
if (TMA == -1) return;//動けないなら終了
MoveTileSub(Tx, Ty, TMA);//動けるなら配列上で動かす
上のような感じでs（殴
これでは見にくいので・・・

おおまかな説明を・・・

ではさっきの関数の
GetTilePoint(Number, &Tx, &Ty);
GetTileMoveAngle(Tx, Ty);
MoveTileSub(Tx, Ty, TMA);
↑代表的な関数たち
大まかな説明をします。

GetTilePoint(Number, &Tx, &Ty);

返り値：void
引数：int Number,int &Tx,int &Ty
説明：一次元配列から引数Numberを見つけてその二次元データ上の座標を返す。
中身↓

```
void NC15Puzzle::GetTilePoint(int Number, int *x, int *y)
{
    int X = 0;//カウント用
    int Y = 0;//カウント用
    for (Y = 0;Y < 4; Y++)//Yを見つける
    {
        for (X = 0;X < 4;X++)//Xを見つける
        {
            //一次元配列上での二次元配列の座標の関係=Y * 横の長さ + Xで求められる
            if (TileData[Y * 4 + X] == Number)
            //↑二次元配列上でのXYの場所にNumberがあるなら
            {
                *x = X;//ポインタを実体参照してXを代入
                *y = Y;//同じくYを代入
                return;//終了
            }
        }
    }
    *x = -1;
    *y = -1;
}
```

GetTileMoveAngle(Tx, Ty);

戻り値：int

引数：int Tx,int Ty

説明：二次元データ上のTx,Tyにあたる場所に空白が隣接しているならその方向を返す。
0＝上、1＝右、2＝下、3＝左、-1＝ない

中身↓（仮想言語）

```
int GetTileMoveAngle(int Tx,int Ty)
{
        4回くりかえす。(int i;i < 4;i++)
        {
                iの値によってある角度に隣接している数値を取得する
                隣接しているところに0(データ上での空白)があるなら
                iを角度として返す。
                もしどの角度にも0がなければ-1を返す。
        }
}
```

MoveTileSub(Tx, Ty, TMA);

返り値：void

引数：int Tx,int Ty,int TMA

説明：二次元データ上のTx,Tyにある数値とTMA(方向)にある数値を入れ替える。

中身↓（仮想言語）

```
void MoveTileSub(int Tx,int Ty,int TMA)
{
        もしTMAが0（移動先の方向が上）なら
        {
                Tx,Tyの数値とTx,Ty + 1の数値を入れ替える
        }
～～～～～～～～～～以下省略～～～～～～～～～～
}
```

これらの関数があれば･･･

１５パズルはできるのです。
ではグラフィック版の中身を一部紹介
次ページ

15パズルヘッダーファイル中身

NG15Puzzle.h*

NewGui15Puzzle　　(グローバル スコープ)

```
#pragma once
//インクルード文
#include"DxLib.h"　//DxLib.hをインクルード
#include <stdlib.h>//乱数を発生させるために
#include <time.h>　//乱数のseed値を時間で取得する

//マクロ定義
#define START_X 192//ナンバータイルの初期描画x座標
#define START_Y 112//ナンバータイルの初期描画y座標
#define LEVEL 100　//シャッフルする回数

//クラス
class NG15Puzzle
{
public:
	//メンバ変数
	int TileHandle[16];//ナンバータイルのグラフィックハンドル
	int TileData[16];　//ナンバータイルの状態
	int CrtGH;　　//クリア後のacongratulations!のグラフィックハンドル
	int TargetNum;　//入力された数値
	int MoveCounter;　//動かした回数
	bool MoveFlag;　//数字が動いているか（仮）
	bool ClearFlag;　//クリアしているかクリア＝true;
	bool ShuffleFlag;
	//メソッド
	NG15Puzzle();　　//コンストラクタ、初期化関数
	void PuzzleMain(void);//メイン関数ここから様々な関数を呼び出す
	void DrawPanel(void);　//バック（ナンバータイルの後ろ）を描画する
	void DrawTiles(void);　//タイルを描画する
	void InputGUI(void);　//ユーザーから入力された値をTargetNumに代入する
	void GetTileLP(int num, int *tx, int *ty);//引数numの配列上の座標をtx,tyに代入する
	void GetTileGP(int num, int *tx, int *ty);//引数numの画面上の座標をtx,tyに代入する
	int GetTileData(int x, int y);　　//配列上での引数x,yにある数値を返す
	void ChangeTilePoint(int x1, int y1, int x2, int y2);//配列上でのx1,y1の値とx2,y2の値を交換する
	void MoveTileSub(int x, int y, int angle);　　//ナンバータイルをChangeTilePointなどを使って移動させる
	void MoveTile(int num);　　//MoveTileSubに必要な値を取得してMoveTileSubを呼び出す
	int GetTileMoveAngle(int x, int y);　　//配列上でのx,yにあたる数値の隣接しているところに空白（0）があれば角度を返す
	void ShuffleTiles(long count);　　//TargetNumに適当な数字をいれてMoveTile関数を呼び出してシャッフルする
	void CheckClear();
};
```

これで終わりです！

次は2048というパズルゲームを作りたいと思います！

質問やアドバイス、提案がありましたら、
お願いします_(._.)_