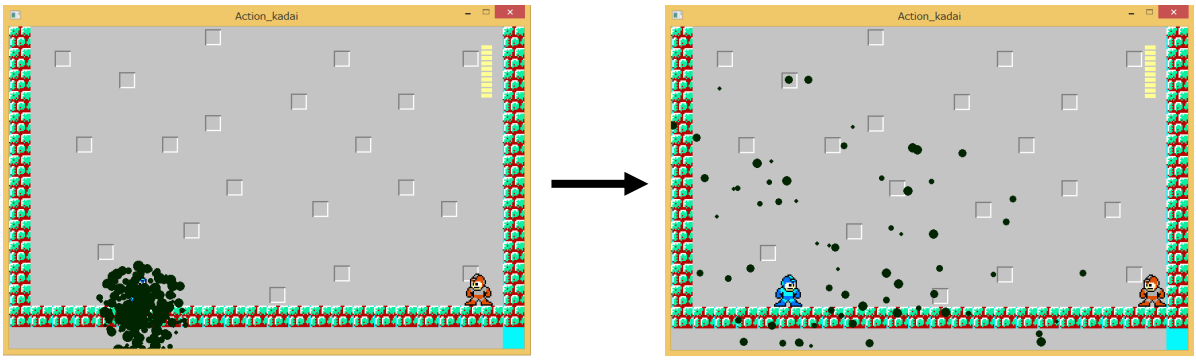
課題 01.爆破エフェクト

1）爆破エフェクトの仕様を決める

■プレイヤーの中心点から沢山の丸いオブジェクトが破裂するイメージ

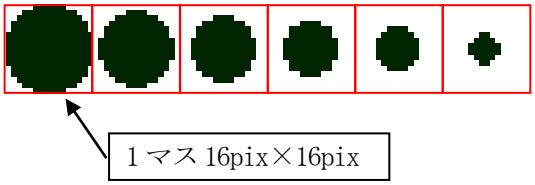


やられた瞬間　　全方向にランダムに飛び散って落下

2）必要なものを準備する

■画像アニメーションを使用して大小の変化を付ける

1)p1tobichiriImage[6]



※LoadDivGraph 命令などで「image/tobichiri.png」ファイルを読み込む

■変数の追加

1)action.h

```
#define TOBICHIRI_MAX 300          // 300個ぐらい

// ----- 飛び散り用
typedef struct {
        XY pos;       // ﾊﾟｰﾂ1個の座標
        XY mov;       // ﾊﾟｰﾂ1個の移動量
        float speed;     // ﾊﾟｰﾂ1個のｽﾋﾟｰﾄﾞ
        float angle;     // ﾊﾟｰﾂ1個の移動方向
        int no;          // ﾊﾟｰﾂのｻｲｽﾞ(ｱﾆﾒｰｼｮﾝ用)
}TOBICHIRI;
```

2)player1.cpp

```
TOBICHIRI p1Dead[TOBICHIRI_MAX];   // 飛び散りパーツを準備

float p1tobichiriAcc;      // パーツの加速度
bool p1tobichiriFlag;      // 飛び散り出現中
float p1angle;             // 計算用
```

3）アルゴリズム

■ライフが0になり、飛び散りが発生していない時（飛び散り発動時）

1)飛び散り弾を全て初期化する ※発動時一度だけ実行

```
// 飛び散りﾊﾟｰﾂをｾｯﾄ
void P1TobichiriSet(void)
{
    p1angle = 0.0f;        // 角度を0度に設定
    p1tobichiriAcc = 0;    // パーツの加速度を初期化

    for(int i = 0; i < TOBICHIRI_MAX; i++){
        p1Dead[i].no = GetRand(5);  // 大小の画像をﾗﾝﾀﾞﾑに設定(0～5)
        p1Dead[i].angle = p1angle;  // 角度を決める(0度から3度ずつずらす)
        p1Dead[i].speed = GetRand(16) + 1;     // 1～16ぐらいで設定
        p1Dead[i].pos.x = p1.pos.x;  // ﾌﾟﾚｲﾔｰの座標X
        p1Dead[i].pos.y = p1.pos.y;  // ﾌﾟﾚｲﾔｰの座標Y
        // 角度を元に移動量を設定(angleを使う)
        p1Dead[i].mov.x = cos((PI/180)*p1angle) * p1Dead[i].speed;
        p1Dead[i].mov.y = sin((PI/180)*p1angle) * p1Dead[i].speed;
        p1angle += 3.0f;    // 3度ずつ回転(次のp1Dead[i].angleに使用する)
    }
    p1tobichiriFlag = true;          // 飛び散り中に設定
}
```

■飛び散りが発生している時

1)更新作業

```
void P1TobichiriUpdate(void)
{
    // 全パーツ移動処理(加速付き)
    for(int i = 0; i < TOBICHIRI_MAX; i++){
        p1Dead[i].pos.x += p1Dead[i].mov.x;
        p1Dead[i].pos.y += p1Deaf[i].mov.y;

        // 移動量は少しずつ減速させていく
        p1Dead[i].speed -= 0.1f;

        // 減速して↓に重力をかけたので移動量を再計算する
        p1Dead[i].mov.x = cos((PI/180)*p1Dead[i].angle) * p1Dead[i].speed;
```

```
            p1Dead[i].mov.y = sin((PI/180)*p1Dead[i].angle) * p1Dead[i].speed
                              + p1tobichiriAcc;
        }
        // 少しずつ↓方向に重力をかけていく為の処理
        p1tobichiriAcc += 3.0f;

        // 全部消えたかチェック
        全部のパーツのY座標がSCREEN_SIZE_Yを超えていれば、全て画面外となる
        総当たりでY座標をチェックして外に出たパーツの数をカウントしていく。
        そのカウントがTOBICHIRI_MAX以上になっていれば全て消えた事になる。
        p1tobichiriFlag = 0;
        // 全部消えたかチェック
        int cnt = 0;
        for(int i = 0; i < TOBICHIRI_MAX; i++){
            if(p1Dead[i].pos.y > SCRENN_SIZE_Y){
                cnt++;
            }
        }

        // 飛び散り終了
        if(cnt >= TOBICHIRI_MAX){
            p1tobichiriFlag = false;
            // ゲームオーバーなどの処理
        }
    }
```

2)描画作業

```
void P1TobichiriDraw()
{
    // 全パーツ描画処理(ｱﾆﾒｰｼｮﾝによるﾊﾟｰﾂのｻｲｽﾞ変更)
    for(int i = 0; i < TOBICHIRI_MAX; i++) {
        DrawGraph(p1Dead[i].pos.x - 8,
                  p1Dead[i].pos.y - 8,
                  p1tobichiriImage[p1Dead[i].no%6], true);
        p1Dead[i].no++;
    }
    DrawString(400 - 40, 200, "2P WIN!", 0xffffff);
}
```