EffectManagerの使用方法

Class の中身はこのようになっている

```
public:
    EffectManager(cocos2d::Layer& layer);
    ~EffectManager();

    void update(float delta);

    void Play(const EffectType& eType, cocos2d::Vec2 pos);
    // ｴﾌｪｸﾄのﾛｰﾄﾞ
    void Load(EffectType eType,int frame,float duration);
    // ｴﾌｪｸﾄｱﾆﾒｰｼｮﾝのｷｬｯｼｭの取得
    const cocos2d::Animation* GetAnimation(const EffectType& eType)
    {
        return effectAnimation_[eType];
    }

    const bool& GetAnimEnd(void)
    {
        return isAnimEnd_;
    }
```

主にこの2つを使用
ここではこの2つの説明。
後はコメントに書いておく。

```
private:
    float animFrame_;
    EffectType type_;
    bool isAnimEnd_;
    cocos2d::Animate* animate_;
    std::map<EffectType, cocos2d::Animation*> effectAnimation_;

    cocos2d::Vec2 pos_;
```

Load 関数の使用方法①:
引数(ロードしたいエフェクトのタイプ、アニメーション総フレーム数、1コマにかかる時間)

ヘッダー側

```
public:
    EffectManager(cocos2d::Layer& layer);
    ~EffectManager();

    void update(float delta);

    void Play(const EffectType& eType, cocos2d::Vec2 pos);
    // エフェクトのロード
    void Load(EffectType eType,int frame,float duration);
    // エフェクトアニメーションのキャッシュの取得
    const cocos2d::Animation* GetAnimation(const EffectType& eType)
    {
        return effectAnimation_[eType];
    }

    const bool& GetAnimEnd(void)
    {
        return isAnimEnd_;
    }
```

Cpp側

```
EffectManager::Load(EffectType eType,int frame,float duration)

std::string effectPath = "";
std::string effectName = "";
switch (eType)
{
case EffectType::PlayerSpawn:
    effectPath = "";
    effectName = "playerSpawn";
    break;
case EffectType::EnemySpawn:
    effectPath = "effect/enemySpawn";

    effectName = "enemySpawn";
    break;
case EffectType::PlayerAttack1:
    effectPath = "";

    effectName = "playerAttack1";
    break;
case EffectType::PlayerAttack2:
    effectPath = "";

    effectName = "playerAttack2";
    break;
case EffectType::Max:
    break;
default:
    break;
}
```

```
// アニメーションキャッシュはシングルトン
AnimationCache* animationCache = AnimationCache::getInstance();

//スプライトシートの準備
auto cache = SpriteFrameCache::getInstance();

// パス指定
cache->addSpriteFramesWithFile(effectPath + ".plist");

// アニメーション画像追加
Animation* animation = Animation::create();

for (int i = 0; i < frame; i++)
{
    auto string = effectName + "%d.png";        // plistの中だからパスじゃない
    auto str = StringUtils::format(string.c_str(), i);
    SpriteFrame* sprite = cache->getSpriteFrameByName(str);

    animation->addSpriteFrame(sprite);
}

// アニメーションの間隔
animation->setDelayPerUnit(duration);

// アニメーション終了後に最初に戻すかどうか
animation->setRestoreOriginalFrame(true);

// 出来たアニメーションをキャッシュに登録
animationCache->addAnimation(animation,effectName);
//animationCache_->addAnimation(animation, actorName + "_" + animName);
// effectAnimation_に登録
effectAnimation_.try_emplace(eType, animation);
```

Load 関数の使用方法②:
引数(ﾛｰﾄﾞしたいｴﾌｪｸﾄのﾀｲﾌﾟ、ｱﾆﾒｰｼｮﾝ総ﾌﾚｰﾑ数、1ｺﾏにかかる時間)

```
EffectManager::Load(EffectType eType,int frame,float duration)

std::string effectPath = "";
std::string effectName = "";
switch (eType)
{
case EffectType::PlayerSpawn:
    effectPath = "";
    effectName = "playerSpawn";
    break;
case EffectType::EnemySpawn:
    effectPath = "effect/enemySpawn";

    effectName = "enemySpawn";
    break;
case EffectType::PlayerAttack1:
    effectPath = "";

    effectName = "playerAttack1";
    break;
case EffectType::PlayerAttack2:
    effectPath = "";

    effectName = "playerAttack2";
    break;
case EffectType::Max:
    break;
default:
    break;
}
```

EffectPath :ﾛｰﾄﾞしたいｴﾌｪｸﾄまでのpath名

e.x) "effect/fire" "effect/wind"

EffectName :ﾛｰﾄﾞしたいｴﾌｪｸﾄの名前

Plistの中身に書いてあるのが
effect0.png effect1.pngだったら
"effect"となる。

あとはﾛｰﾄﾞしたいｴﾌｪｸﾄのﾀｲﾌﾟの
switch case文で分岐させている。

GameSceneのコンストラクタの中身に記入

```
// ここからｴﾌｪｸﾄ描画する大本の追加
effectManager_ = std::make_shared<EffectManager>(*layer_[static_cast<int>(zOlder::FRONT)]);
effectManager_->Load(EffectType::EnemySpawn, 19, 0.08f);
effectManager_->scheduleUpdate();
```

上記の中で、この関数の呼び出しの時に、
自分が追加したｴﾌｪｸﾄのﾀｲﾌﾟをﾛｰﾄﾞするよ
うに記入。

```
effectManager_->Load(EffectType::EnemySpawn, 19, 0.08f);
```

因みにenumはこんな感じ

```
enum class EffectType
{
    PlayerSpawn,
    EnemySpawn,
    PlayerAttack1,
    PlayerAttack2,
    Max
};
```

必要に応じて追加してくれればOK!

因みに因みに、Effectを新たに追加したら
Resource/Effect の中にそのまま入れてもらえればOKです。

Play 関数の使用方法:
引数(再生させたいｴﾌｪｸﾄのﾀｲﾌﾟ、ｴﾌｪｸﾄのﾎﾟｼﾞｼｮﾝ)

ヘッダー側

```
public:
    EffectManager(cocos2d::Layer& layer);
    ~EffectManager();

    void update(float delta);

    void Play(const EffectType& eType, cocos2d::Vec2 pos);
    // ｴﾌｪｸﾄのﾛｰﾄﾞ
    void Load(EffectType eType,int frame,float duration);
    // ｴﾌｪｸﾄｱﾆﾒｰｼｮﾝのｷｬｯｼｭの取得
    const cocos2d::Animation* GetAnimation(const EffectType& eType)
    {
        return effectAnimation_[eType];
    }

    const bool& GetAnimEnd(void)
    {
        return isAnimEnd_;
    }
```

ｴﾌｪｸﾄを再生させたいﾀｲﾐﾝｸﾞで
一回のみの呼び出しでOK！

後は関数の中のrunActionが
勝手に回ってくれる。
(ﾎﾟｼﾞｼｮﾝ関係はまだ未実装な
ので実装する度にこちらも更新
します。)

Cpp側

```
void EffectManager::Play(const EffectType& eType, cocos2d::Vec2 pos)
{
    pos_ = pos;
    type_ = eType;
    animFrame_ = 0.0f;
    isAnimEnd_ = false;
    animate_ = Animate::create(effectAnimation_[eType]);
    auto action = Repeat::create(Animate::create(effectAnimation_[eType]), 1);
    runAction(action);
}
```