リソース管理

ゲーム制作を行うにあたり、
画像、モデル、エフェクト、サウンドなど、数多くのリソースを
扱っていく必要があります。

そのたくさんあるリソースを、いちいちロードしたり、
int型変数にハンドルを保存したり、使い終わったらメモリ解放したりと
手間がかかります。

これらのリソースを一括して管理し、
情報の集約や、メモリの自動解放を行った方が保守性が高くなります。

ResourceManagerの設計

- シングルトンで設計

- Init関数で、リソースごとの読込パスや種別を全て保存しておく。
  以降、リソース種別を渡すだけで、ロードができる。
  (Init時は、ロードしない)

- 画像は、一度ロードしたら以降、同じハンドルIDを返す

- モデルも同様であるが、
  複数体のモデル制御を行う場合は、複製ロードLoadModelDuplicateを
  使用する

- ロードされたリソースは、シーン遷移時に、全て解放する

```
void SceneManager::DoChangeScene(SCENE_ID sceneId)
{

    // リソースの解放
    ResourceManager::GetInstance().Release();

    // シーンを変更する
    sceneId_ = sceneId;
```

- 1つ1つのリソース情報は、Resourceクラスで管理する

ResourceManagerの使い方

例：TitleSceneで2D画像を読み込む

手順①　ResourceManager.hのenum class SRCにリソース名を登録する

```
// リソース名
enum class SRC
{
    TITLE,
};
```

手順②　ResourceManagerのInit関数で、
resourcesMap_に、リソース名、リソース種別、読み込み先のパスを登録する

```
// タイトル画像
res = new RES(RES_T::IMG, PATH_IMG + "Title.png");
resourcesMap_.emplace(SRC::TITLE, res);
```

※ モデルだと、RES_T::IMGがRES_T::MODELに変わる
現在対応しているの以下のリソース種類。
サウンドは、自分で拡張してみましょう。

```
// リソースタイプ
enum class TYPE
{
    NONE,
    IMG,          単体画像
    IMGS,         複数画像(LoadDivGraph)
    MODEL,        モデル
    EFFEKSEER     エフェクト(EFFEKSEER)
};
```

手順③　　リソースを使用したい場面で、ロードを行う

```
void TitleScene::Init(void)
{

    // 画像読み込み
    imgTitle_ = resMng_.Load(ResourceManager::SRC::TITLE).handleId_;

    ～ 省略 ～

}
```

【要件】

TitleSceneに、タイトル画像と、PushSpace画像を描画しましょう。

タイトル画像	Title.png
PushSpace画像	PushSpace.png

【目標】