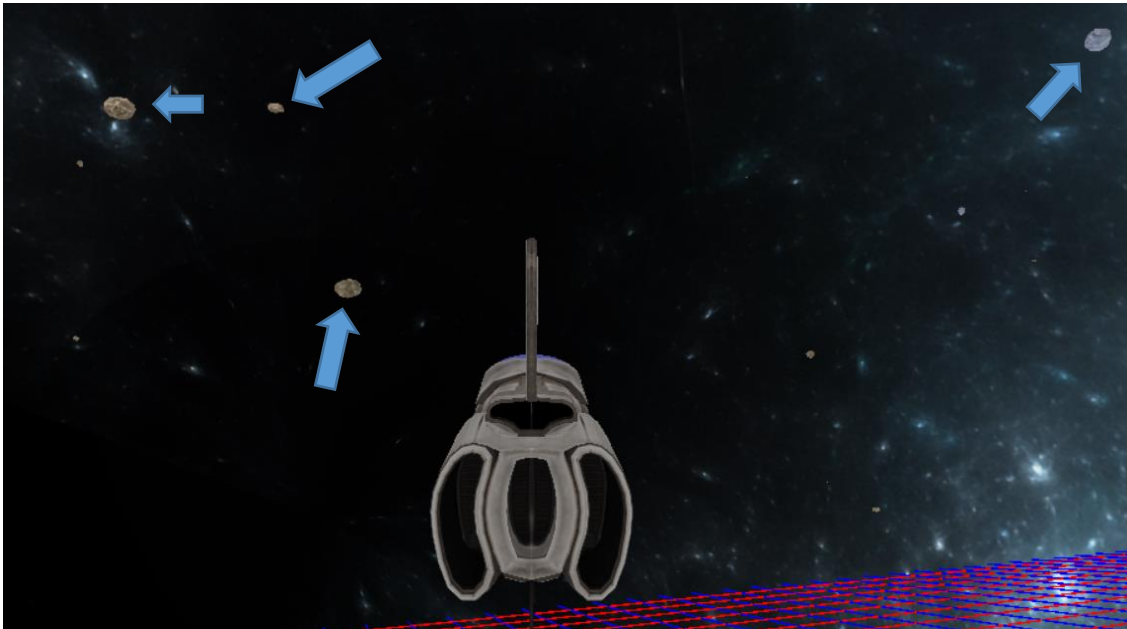


スペースデブリ

自機の自由旋回、バネ付き追従カメラを実装して、宇宙を駆け回れるようになりましたが、イマイチ物足りません。

周りにオブジェクトがありませんので、自機が移動している感があまり感じられないのです。

自機のスピードを考慮すると、かなり広いマップを用意する必要がありそうですが、ここでは、簡単にマップオブジェクト生成して簡易的ではありますが、宇宙の雰囲気作りをしていきたいと思います。



岩のモデルを大量にロードしていきます。
通常、モデルをロードする時は、MVILoadModelを使用しますが、同じモデルをロードする際に、外部ファイルからモデル情報を読み込んで、DxLib用のモデル情報を構築するよりも、既に内部に持っているモデル情報を複製する方が、遥かに処理速度が早いので、同じモデルを読み込む場合は、

MVIDuplicateModel

を使用します。

数百メガのメモリの使用してしまう場合がありますので、
メモリの解放、MVDeleteModelも大切になってきます。

皆さんを信用していないわけではないのですが、
確実にメモリを解放するように、
リソース(画像やモデル)を管理する専用のクラスを作成していますので、
そちらを使用してください。
(メモリ管理を既に構築されていたり、キチンできる方は使用しなくてOKです)

ResourceManagerクラス

Resource Load(SRC src);

→ 画像やモデルをロードする時に使用してください。
SRCは今回のプロジェクトで使用するリソースの種類です。

```
enum class SRC
{
    SHIP_EXPLOSION,      機体の爆発エフェクト
    SHOT_EXPLOSION,      弾の爆発エフェクト
    SPEECH_BALLOON,      吹き出し
    SHOT_MODEL,           弾
    TURRET_STAND,         砲台
    TURRET_GUN,           砲身
    ROCK01,               背景岩 1
    ROCK02,               背景岩 2
};
```

int LoadModelDuplicate(SRC src);

→ 前述の通り、モデルを複製する場合は、こちらを使用してください。

Resourceクラス

```
int mHandleId;  
int* mHandleIds;
```

Loadの返り値で渡ってきたResourceのmHandleIdを参照すれば、
ハンドルIDを取得できます。

LoadDivGraphなどの分割読み込みの場合は、
mHandleIdsポインタ変数を参照してください。

スペースデブリの話に戻ります。


一度に、ステージで使用するマップ全体のデブリ(岩)を読み込むと
処理時間も大変なことになりますので、
皆さん得意なマップ座標を用いて、段階的にロードしていきたいと思います。

仮に、


1マップあたりのサイズを2000

1マップあたりに生成する岩の数を30

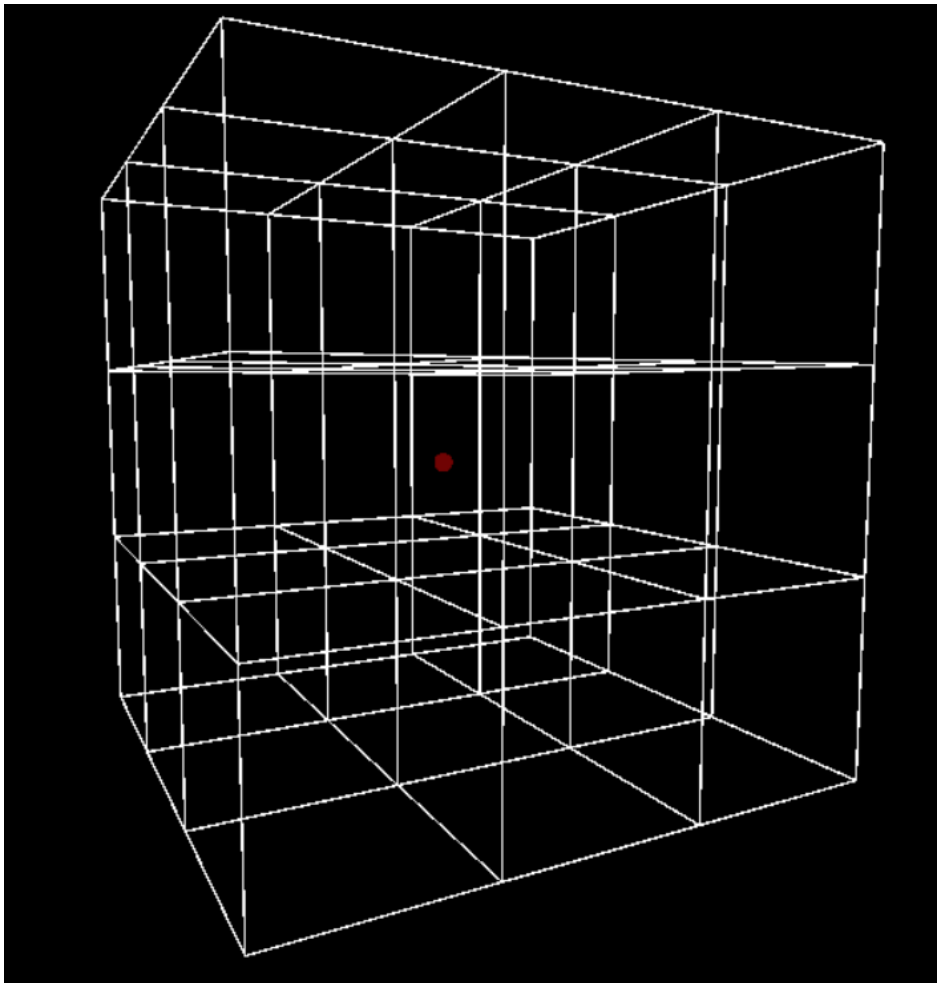
とした場合

- 2000 - - 2000 - - 2000 - - 2000 - - 2000 - - 2000 - - 2000 -						
		30個生成	30個生成	30個生成		
		30個生成	 30個生成	30個生成		
		30個生成	30個生成	30個生成		

操作プレイヤーのマップ座標に応じて、周囲のオブジェクトを生成していく。

- 2000 - - 2000 - - 2000 - - 2000 - - 2000 - - 2000 - - 2000 -						
		30個	30個	30個	30個生成	
		30個	30個		30個生成	
		30個	30個	30個	30個生成	

わかりやすいように2次元の表を使いましたが、
当然、マップは3次元で構築します。



この1つのマップごとに、位置や岩の角度や大きさをランダムに生成すれば、デブリの完成です。

注意して貰いたいのが、マップ座標をワールド座標に変換した後、その座標を中心(上図の赤丸)として、ランダムに生成してください。

	▽			
				▽
		●		
▽			▽	

初期座標が、 $\{0, 0, 0\}$ のため、
偏った配置になってしまいます。

可能であれば、
デブリがカメラの視野範囲から外れた場合、
処理や描画を行わないようにしましょう。