モデル用コライダの作成

モデルは、DxLibの方で衝突情報の構築を行ってくれますので、
モデルのハンドルIDされ管理できればＯＫです。

ColliderModel.h

```
#pragma once
#include "ColliderBase.h"

class ColliderModel : public ColliderBase
{

public:

    // コンストラクタ
    ColliderModel(TAG tag, const Transform* follow);

    // デストラクタ
    ~ColliderModel(void) override;

protected:

    // デバッグ用描画
    void DrawDebug(int color) override {};

};
```

ColliderModel.cpp

```
#include "ColliderModel.h"

ColliderModel::ColliderModel(TAG tag, const Transform* follow)
    :
    ColliderBase(SHAPE::MODEL, tag, follow)
{
}

ColliderModel::~ColliderModel(void)
{
```

```
}
```

Stage.h

```
class Stage : public ActorBase
{

public:

    // 衝突判定種別
    enum class COLLIDER_TYPE
    {
        MODEL = 0,
        MAX,
    };
```

Stage.cpp

```
void Stage::InitCollider(void)
{

    // DxLib側の衝突情報セットアップ
    MV1SetupCollInfo(transform_.modelId);

    // モデルのコライダ
    ColliderModel* colModel =
        new ColliderModel(ColliderBase::TAG::STAGE, &transform_);
    ownColliders_.emplace(static_cast<int>(COLLIDER_TYPE::MODEL), colModel);

}
```

これで、それぞれのActorがColliderを保持している状態になりました。