ゲームエフェクトセミナー

第３部

BISHAMONの導入と実践

マッチロック株式会社

後藤　誠

第３部
BISHAMONの導入と実践

- 3-1：インストールしてみる
- 3-2：サンプルを動かしてみる
- 3-3：サンプルの構成について
- 3-4：STGに導入してみる

MATCHLOCK

3-1：インストールしてみる

■ 必要なものはなに？

・Visual Studio C++ 2008(or 2010)
http://www.microsoft.com/ja-jp/dev/express/default.aspx

・DirectX９ SDK
http://www.microsoft.com/en-us/download/details.aspx?id=6812S

・BISHAMON Personal SDK for DirectX９
http://www.matchlock.co.jp/store/index.php/bm-personal-sdk-dx9-1-0.html

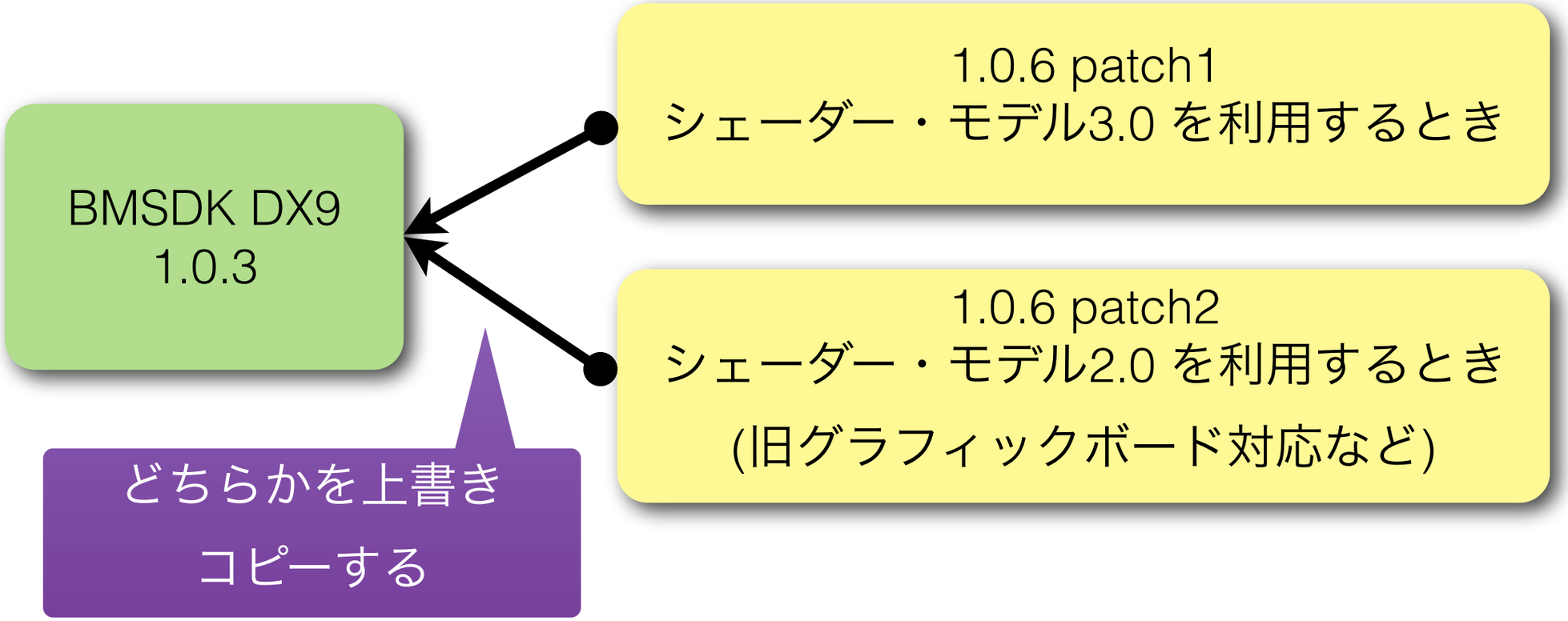
・覚悟とか、やる気とか、根性とか…
「ジョジョの奇妙な冒険」、「ONE PIECE」、「スラムダンク」…

3-1：インストールしてみる

■ BISHAMON SDKの中身はどうなってるの？

BISHAMON Personal SDK for DirectX9
最新版は1.0.6 （1.0.3 に修正パッチを追加）になります。
適応するパッチは、シェーダー・モデルの対応バージョンによって異なります。

- 1.0.3 + 1.0.6 patch1 → SM3.0 用(シェーダー・モデル3.0)
- 1.0.3 + 1.0.6 patch2 → SM2.0 用(シェーダー・モデル2.0)

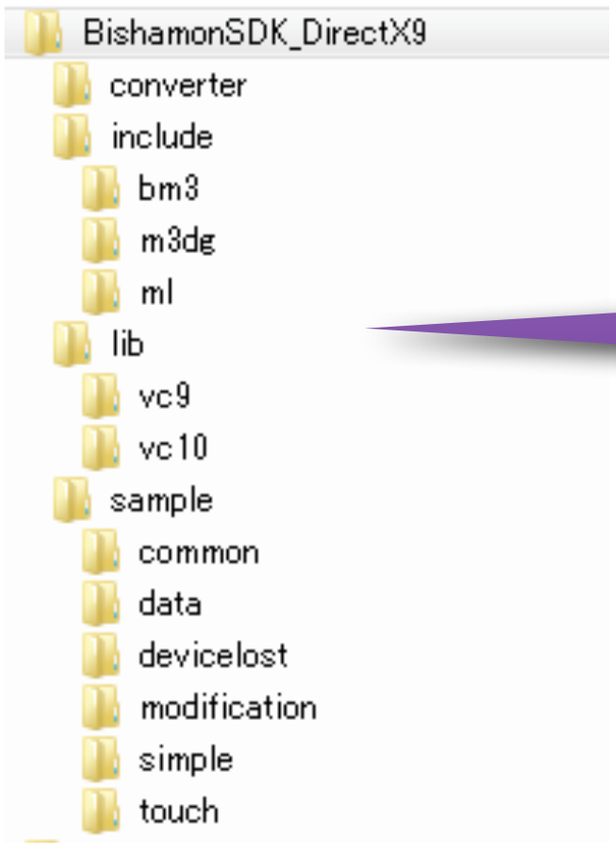


MATCHLOCK

3-1：インストールしてみる

■ BISHAMON SDKはどこに解凍すればいいの？

- 特に制限はありません。
- 適切な場所へ解凍、コピーしてください。



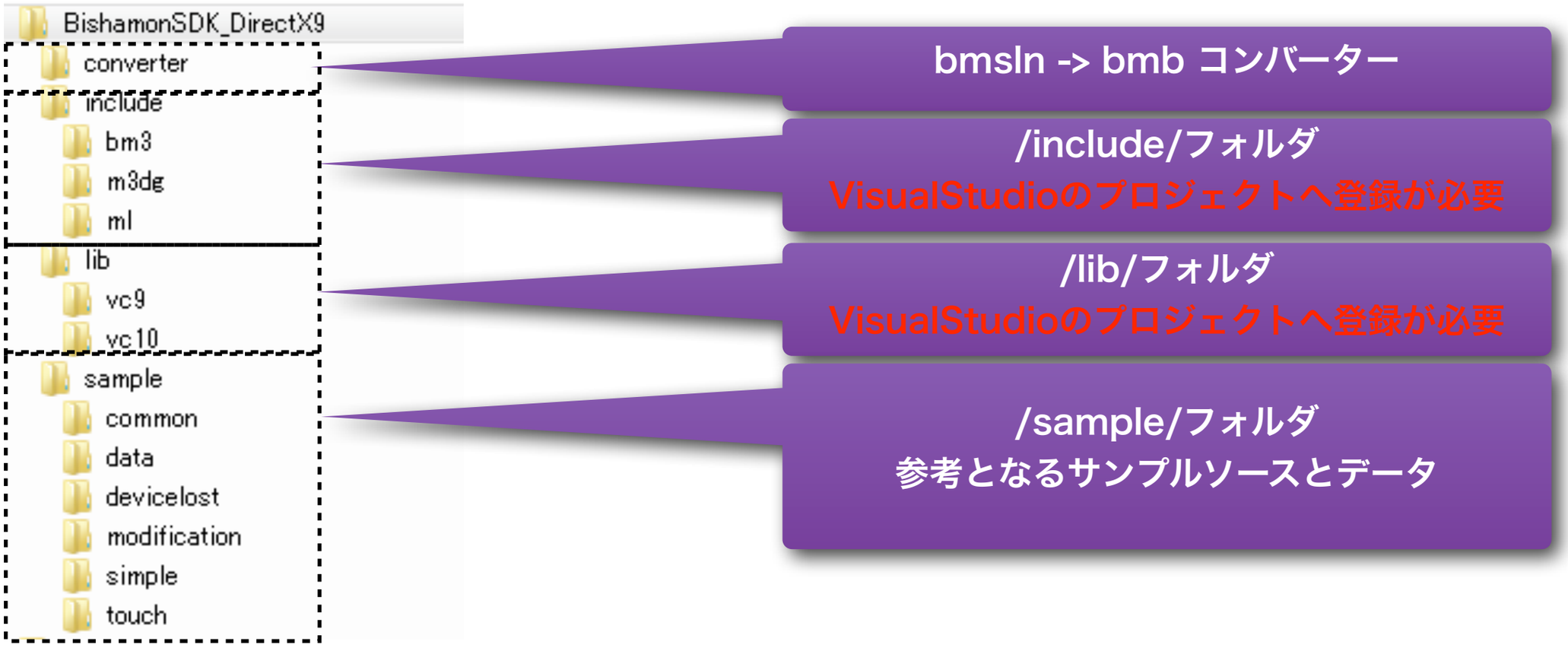
解凍すると、このようなフォルダ構成で
展開されます。

MATCHLOCK

3-1：インストールしてみる

■ BISHAMON SDKの中身はどうなってるの？

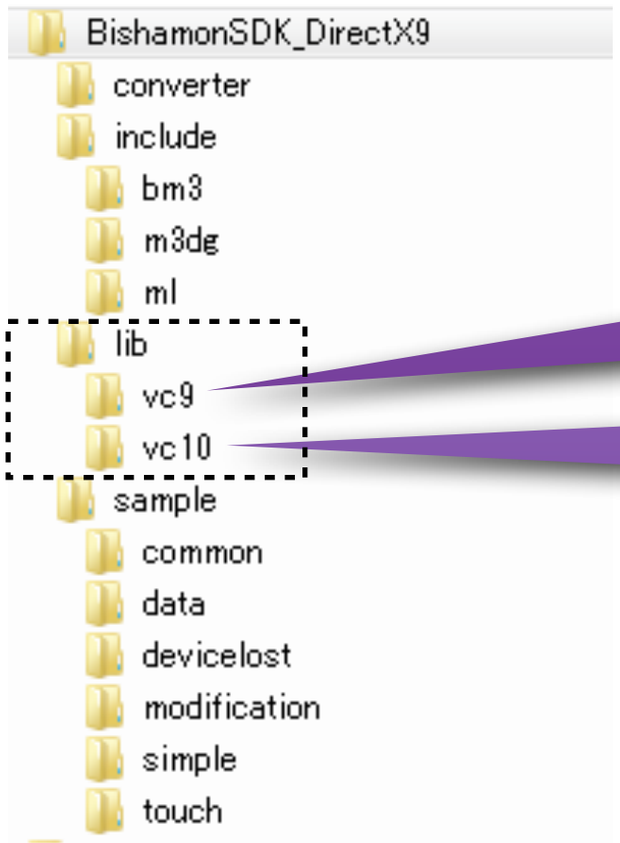
・/include/ と /lib/ とコンバータ、サンプル



3-1：インストールしてみる

■ VS2008とVS2010どちらでも使えるの？

・使えます！！下記の設定変更が必要です。



VS2008 を利用するときは、
/lib/vc9/フォルダを
プロジェクトの追加ライブラリーへ登録

VS2010 を利用するときは、
/lib/vc10/フォルダを
プロジェクトの追加ライブラリーへ登録

注意：VS2012 はまだ未対応です

MATCHLOCK

3-1：インストールしてみる

■ マニュアルはどこにあるの？

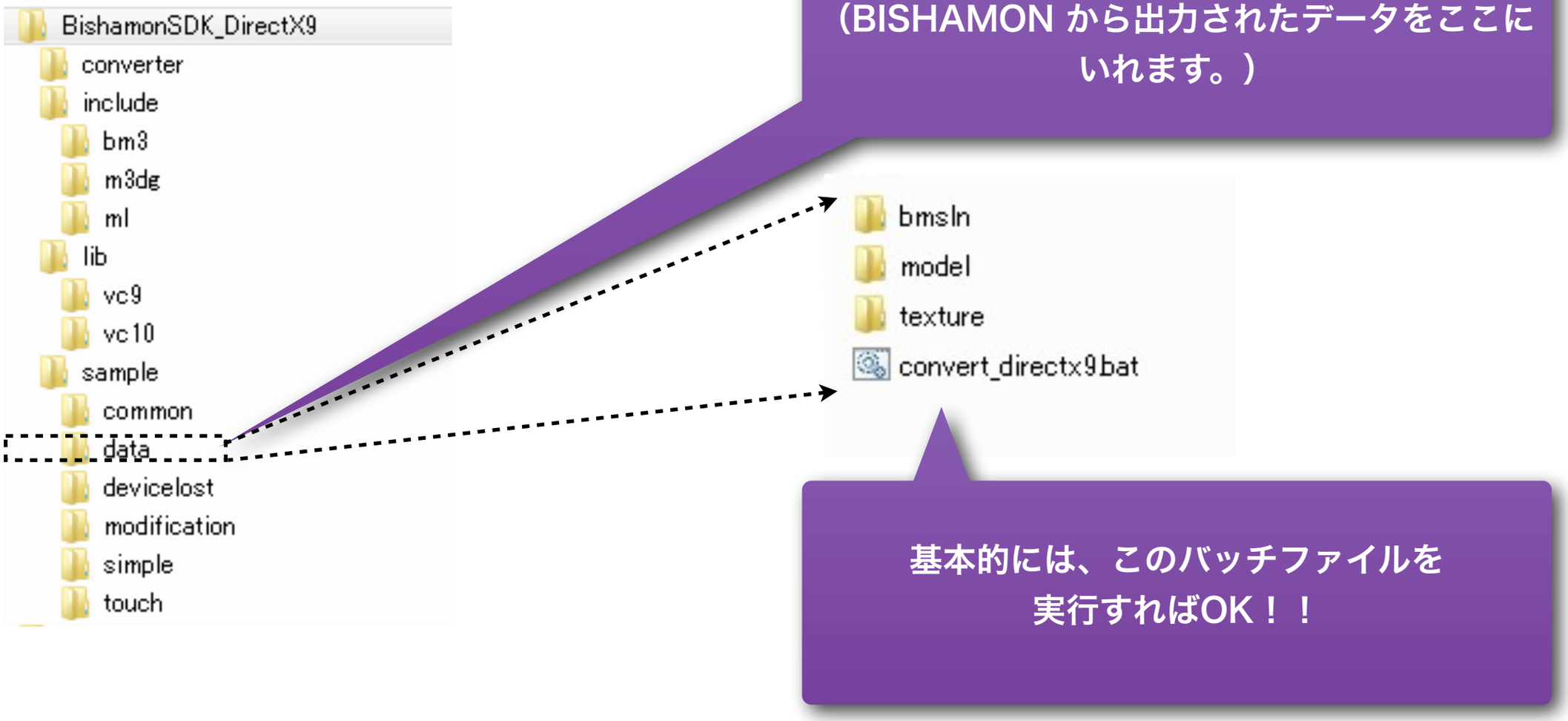
・オンライン・マニュアルで提供しています。

http://www.matchlock.co.jp/products/document/sdk_document/directx9/

MATCHLOCK

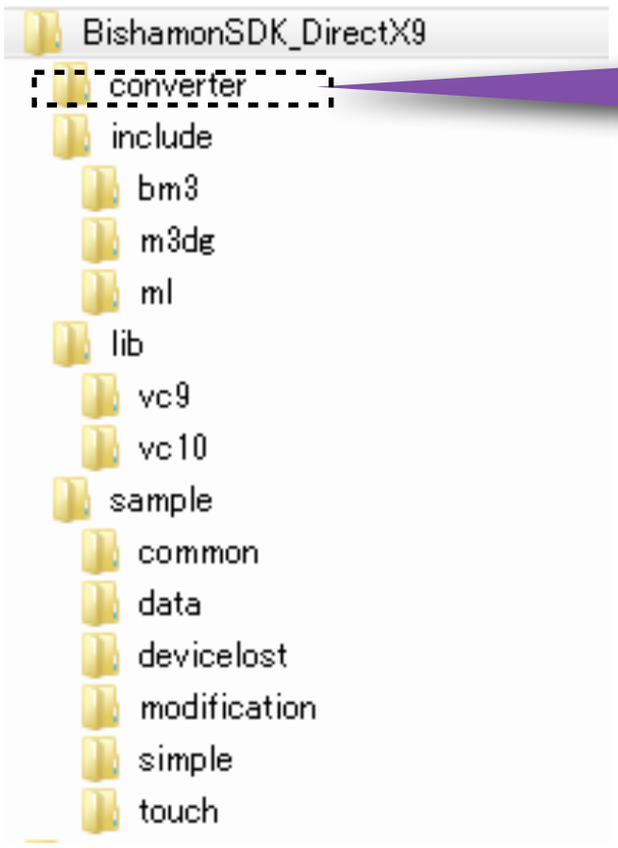
3-1：インストールしてみる

■ バイナリーコンバート（bmsln->bmb）してみる



3-1：インストールしてみる

■ バイナリーコンバート（bmsln->bmb）してみる



注意）

バイナリーコンバートには

有効期限内のライセンス・ファイル

bm.lic

がフォルダ内に必要です！！！

…ん？…つまり…

MATCHLOCK

3-1：インストールしてみる

■ バイナリーコンバート（bmsln->bmb）してみる

http://www.matchlock.co.jp/store/

ですよね～ (^_^;

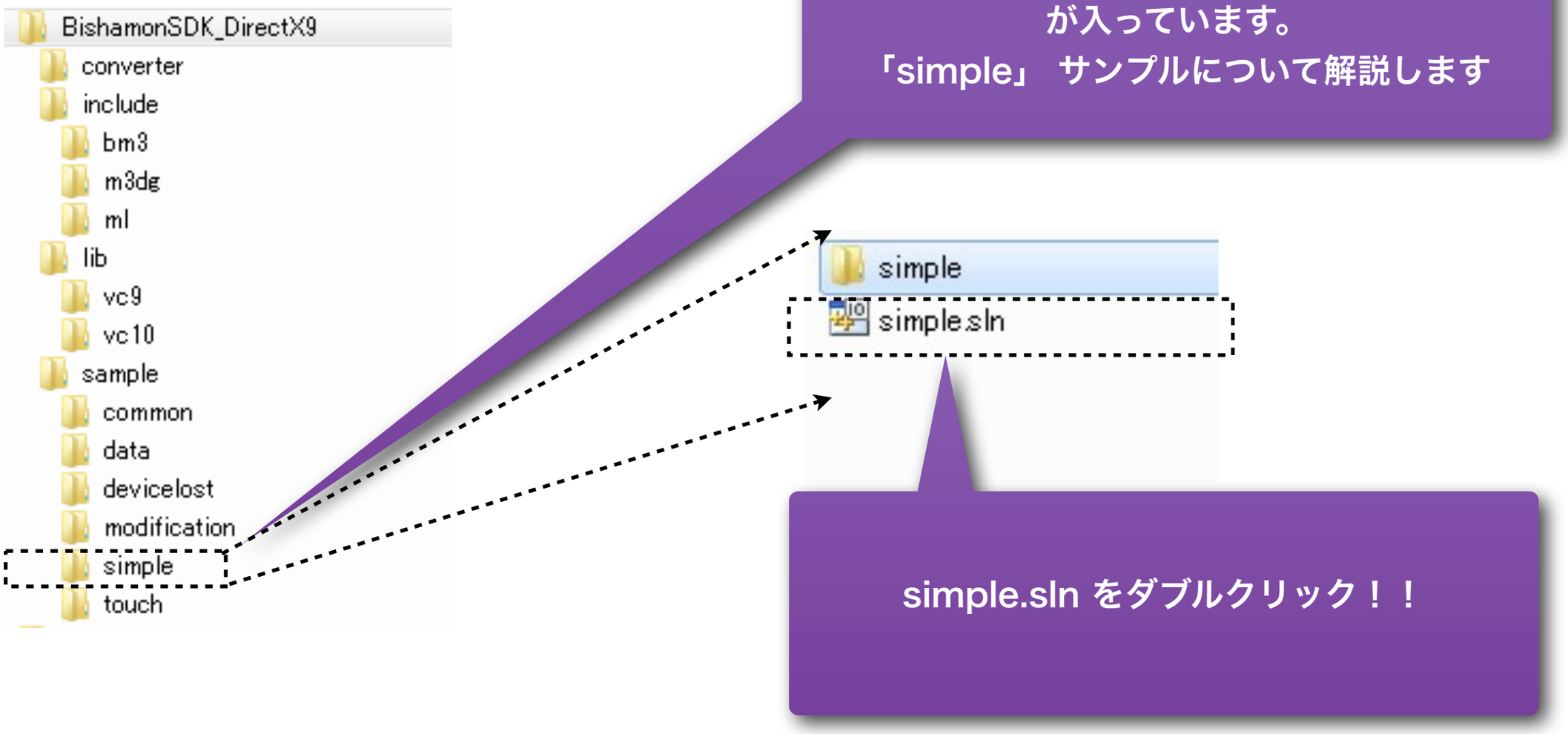
MATCHLOCK

第３部
BISHAMONの導入と実践

- 3-1：インストールしてみる
- 3-2：サンプルを動かしてみる
- 3-3：サンプルの構成について
- 3-4：STGに導入してみる

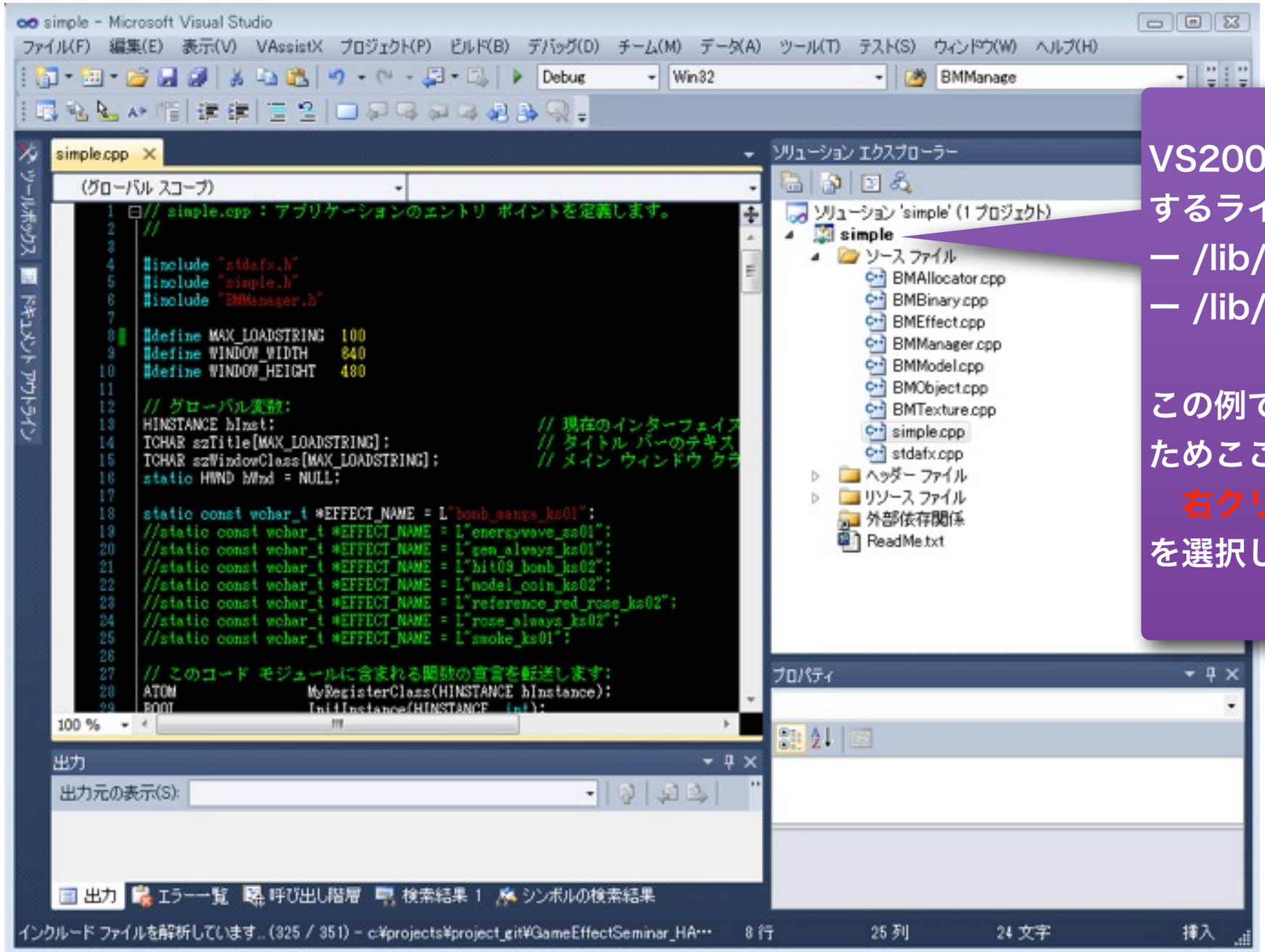
3-2：サンプルを動かしてみる

■ サンプルをコンパイルするにはどうするの？



3-2：サンプルを動かしてみる

■ サンプルをコンパイルするにはどうするの？



VS2008 と VS2010 では、リンクするライブラリーが違います。

ー /lib/vc9/ … VS2008

ー /lib/vc10/ … VS2010

この例では、VS2010 を利用しているためここを

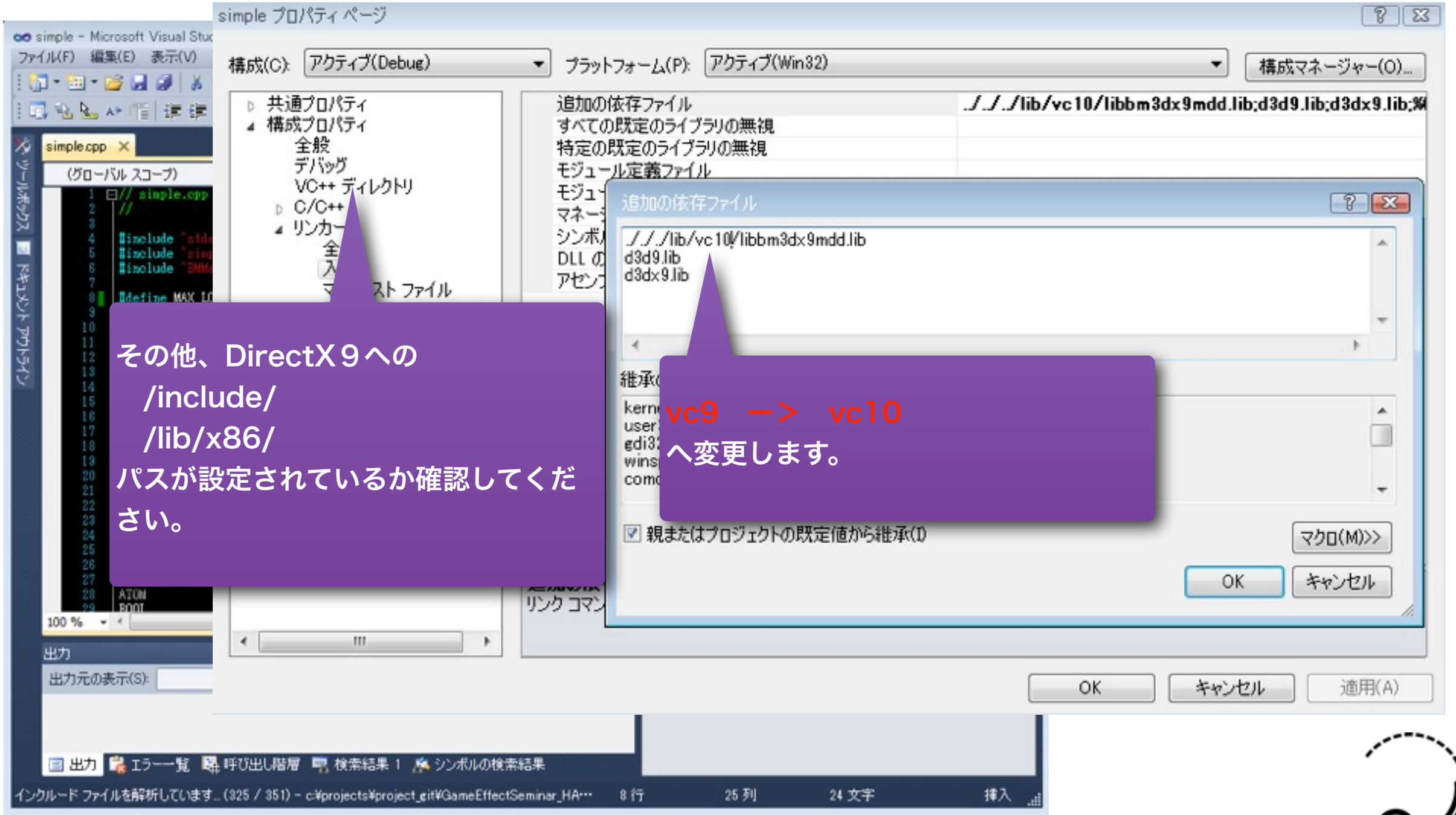
右クリック ー> 「プロパティ」

を選択します。

MATCHLOCK

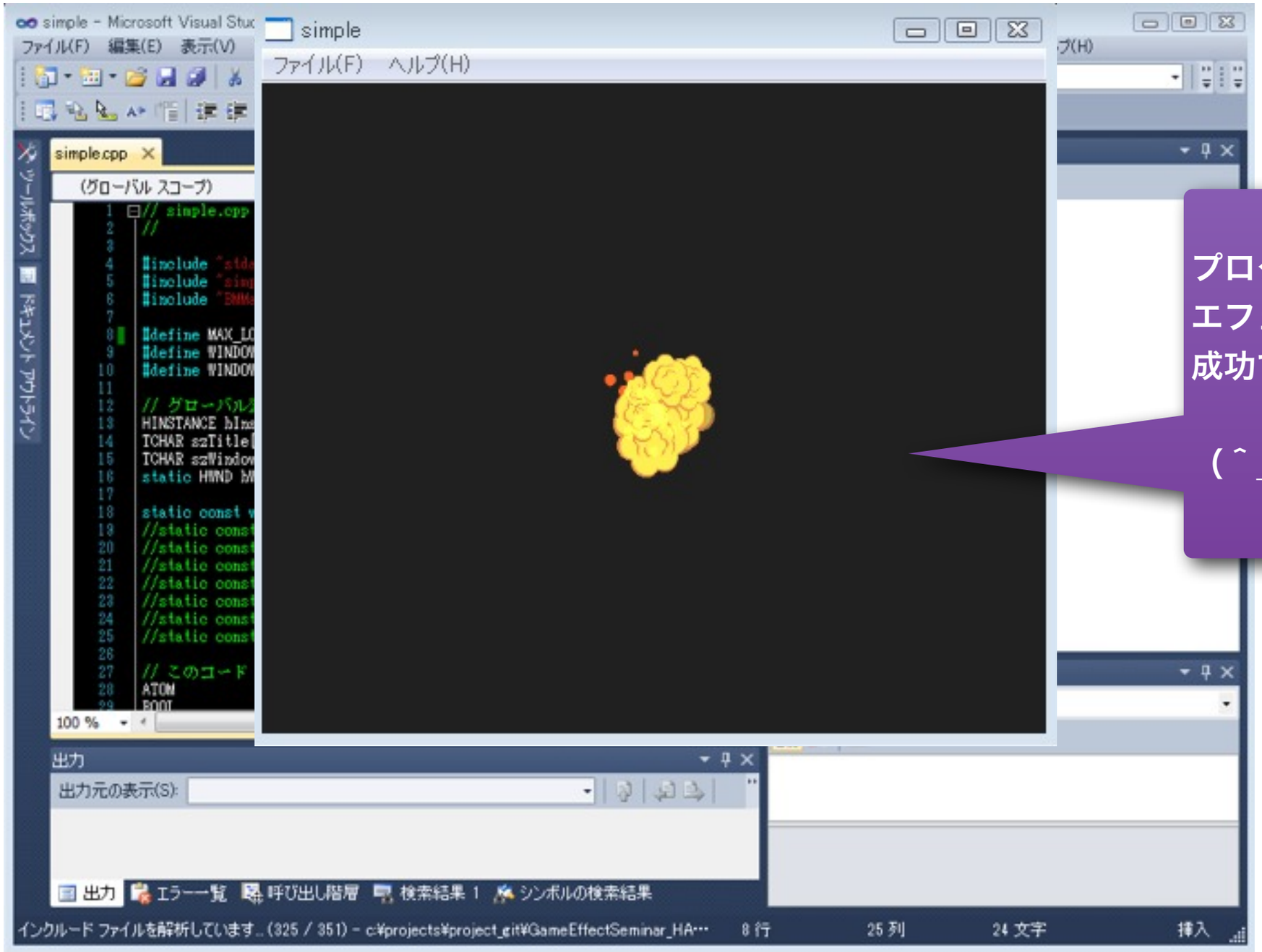
3-2：サンプルを動かしてみる

■ サンプルをコンパイルするにはどうするの？



3-2：サンプルを動かしてみる

■ サンプルをコンパイルするにはどうするの？



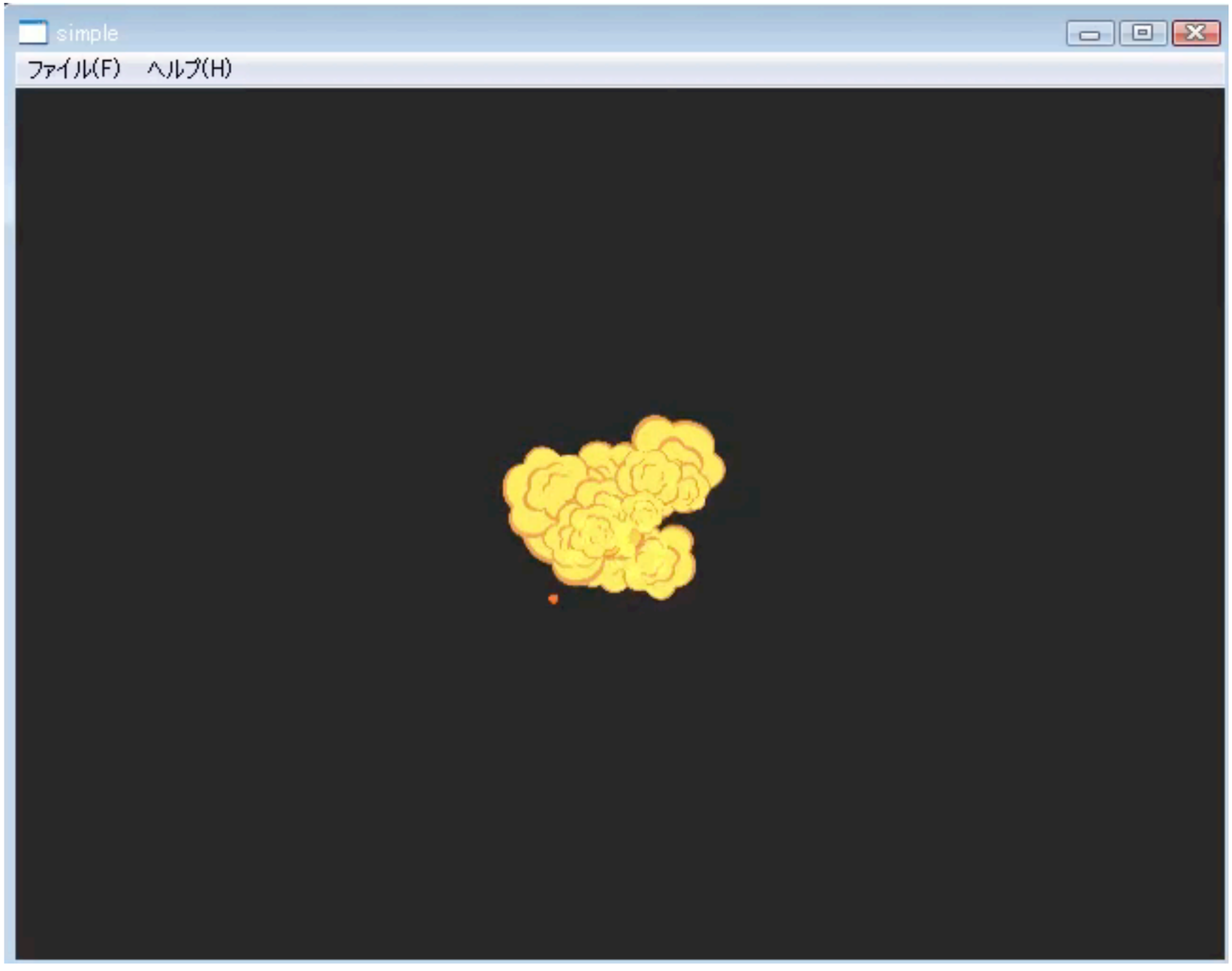
プログラムを実行して、このように
エフェクトが確認できれば、
成功です！！

（＾__＾；　ふ～

MATCHLOCK

3-2：サンプルを動かしてみる

■ サンプルをコンパイルするにはどうするの？



3-2：サンプルを動かしてみる

■ 他のエフェクトも確認できるの？

できます！「simple.cpp」を開いてください！

```
18 static const wchar_t *EFFECT_NAME = L"bomb_manga_ks01";
19 //static const wchar_t *EFFECT_NAME = L"energywave_ss01";
20 //static const wchar_t *EFFECT_NAME = L"gem_always_ks01";
21 //static const wchar_t *EFFECT_NAME = L"hit09_bomb_ks02";
22 //static const wchar_t *EFFECT_NAME = L"model_coin_ks02";
23 //static const wchar_t *EFFECT_NAME = L"reference_red_rose_ks02";
24 //static const wchar_t *EFFECT_NAME = L"rose_always_ks02";
25 //static const wchar_t *EFFECT_NAME = L"smoke_ks01";
```

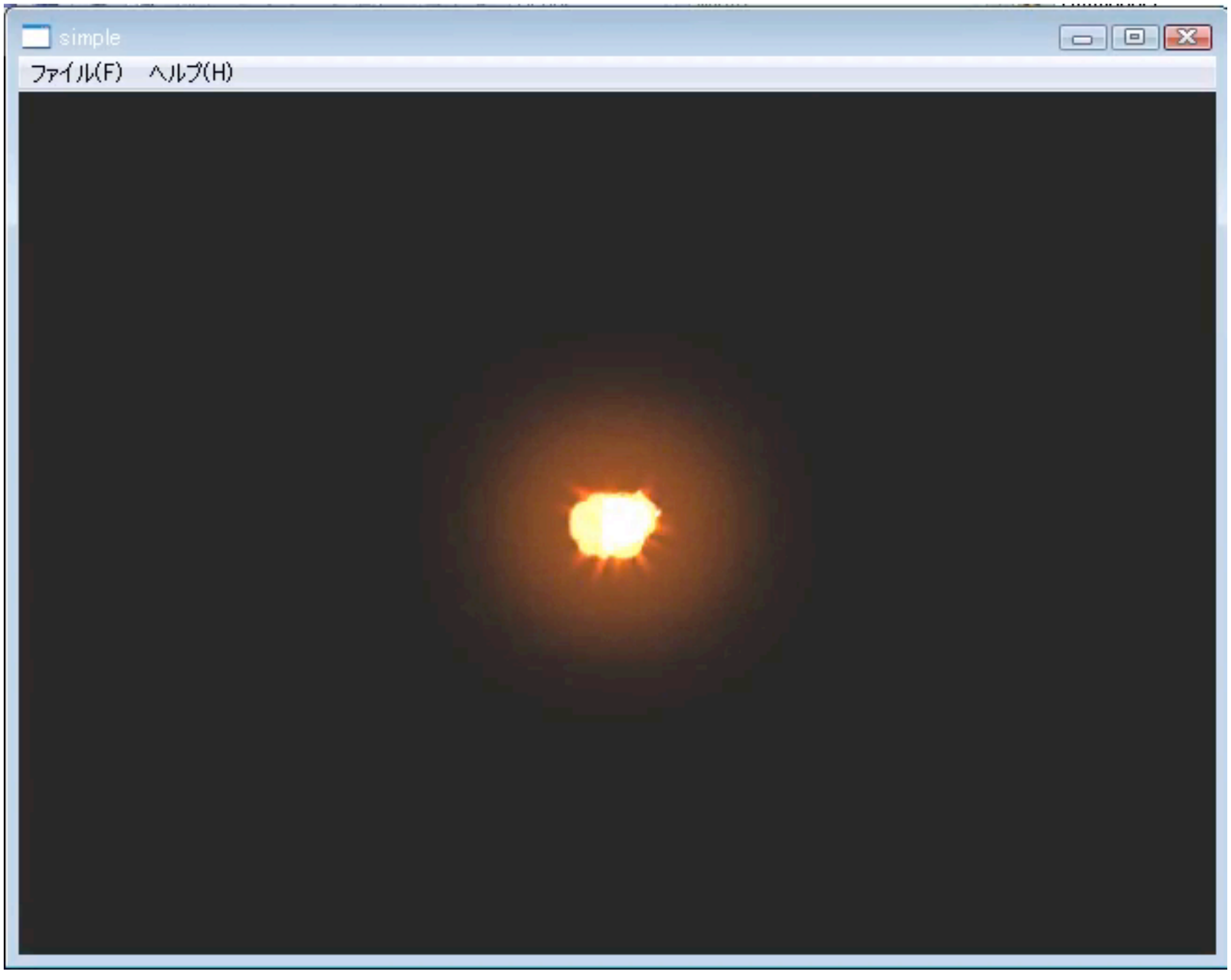
別のエフェクト名を指定するようにします。

```
18 //static const wchar_t *EFFECT_NAME = L"bomb_manga_ks01";
19 //static const wchar_t *EFFECT_NAME = L"energywave_ss01";
20 //static const wchar_t *EFFECT_NAME = L"gem_always_ks01";
21 static const wchar_t *EFFECT_NAME = L"hit09_bomb_ks02";
22 //static const wchar_t *EFFECT_NAME = L"model_coin_ks02";
23 //static const wchar_t *EFFECT_NAME = L"reference_red_rose_ks02";
24 //static const wchar_t *EFFECT_NAME = L"rose_always_ks02";
25 //static const wchar_t *EFFECT_NAME = L"smoke_ks01";
```

MATCHLOCK

3-2：サンプルを動かしてみる

■ 他のエフェクトも確認できるの？

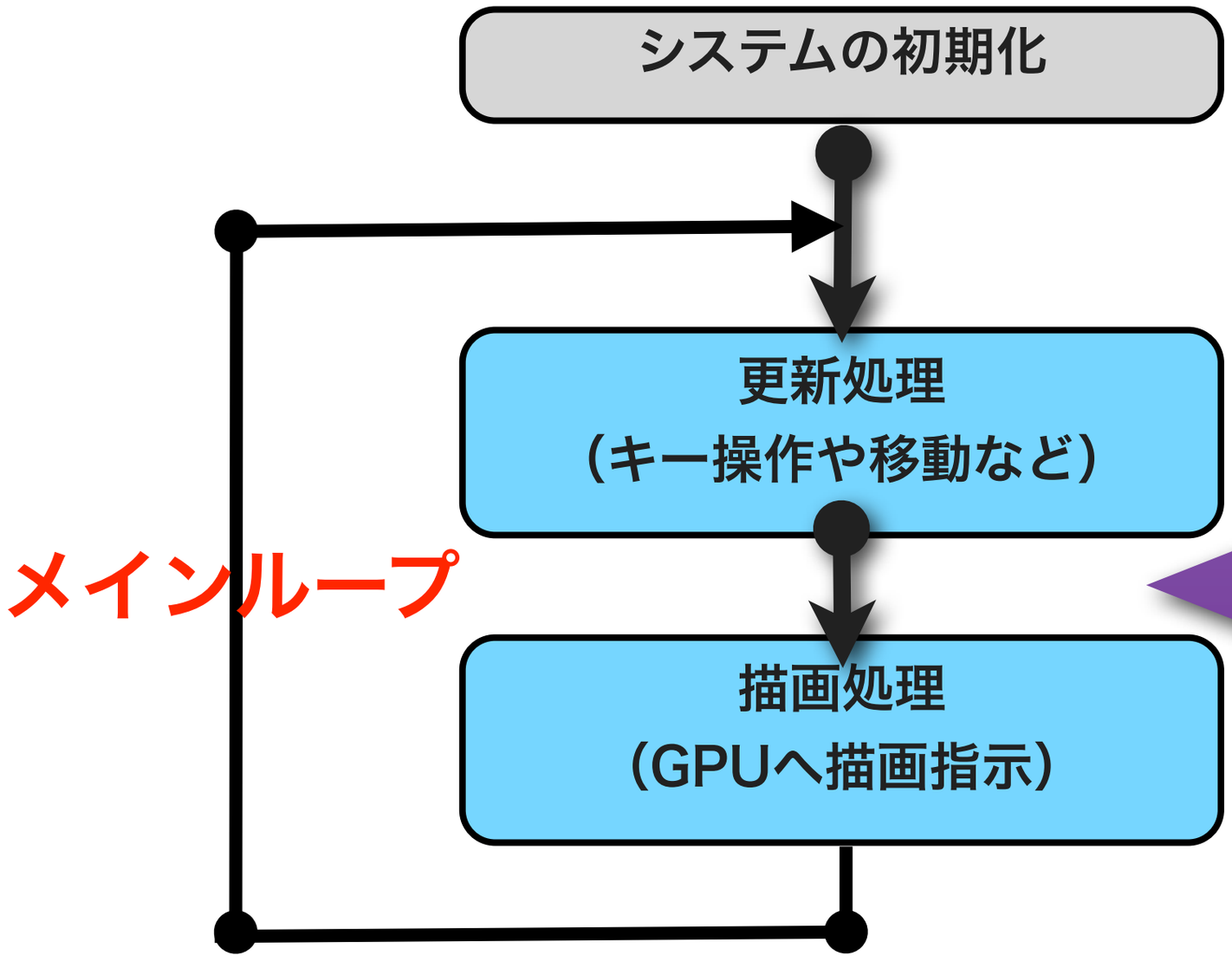


第３部
BISHAMONの導入と実践

- 3-1：インストールしてみる
- 3-2：サンプルを動かしてみる
- **3-3：サンプルの構成について**
- 3-4：STGに導入してみる

3-3：サンプルの構成について

■ 「simple.cpp」はどういう構造なの？



Window更新時など、描画のみ複数回呼ばれたり、逆に更新のみ行なって描画しないことがあるため、「更新」と「描画」がわかれているのが一般的。

3-3：サンプルの構成について

■ 「simple.cpp」はどういう構造なの？

Window の初期化

DirectX の初期化

BISHAMON の初期化
BM EFFECT の読込＆生成

BM EFFECT の更新

BM EFFECT の描画
（カメラ情報が必要）

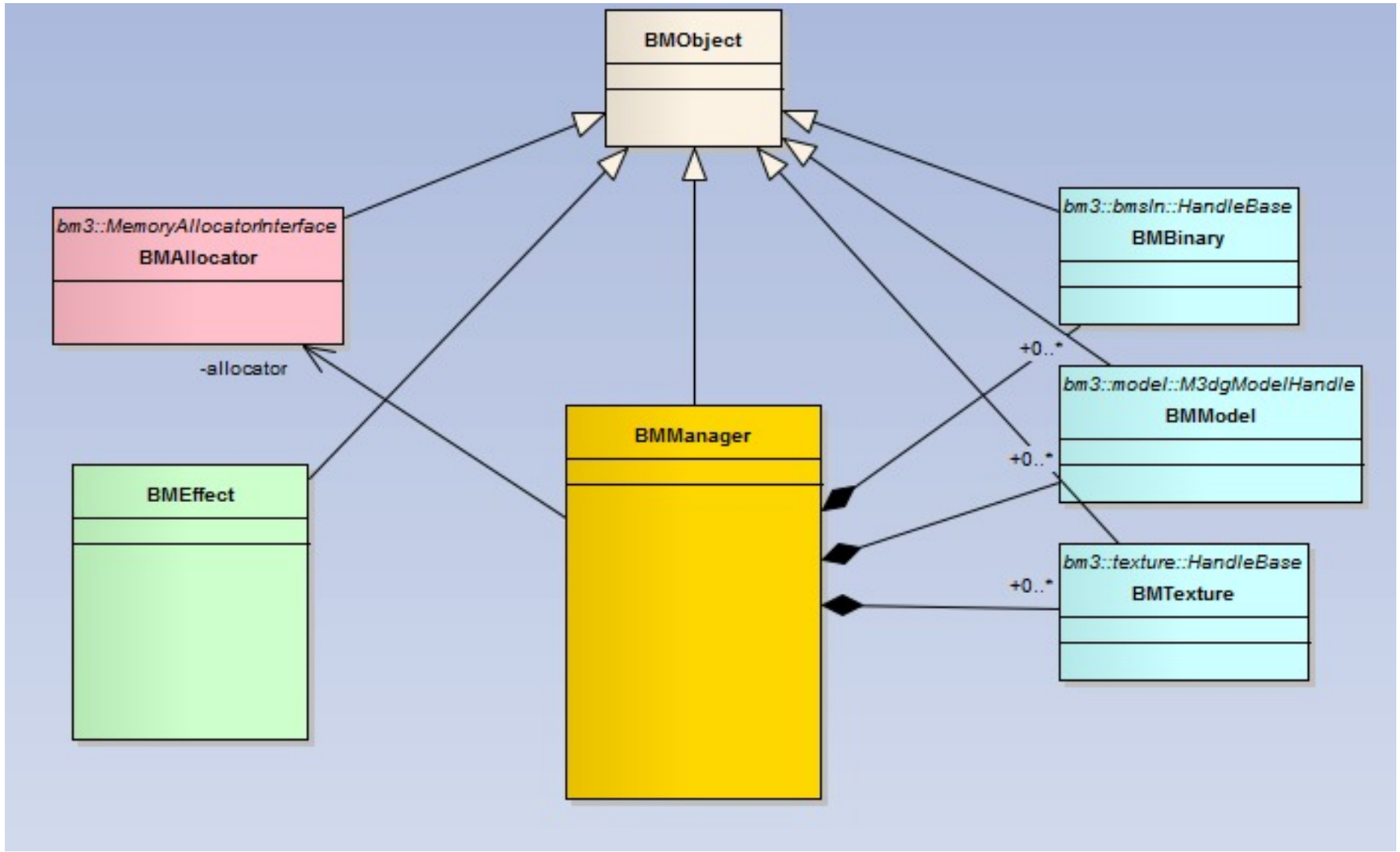
メインループ

```
82      BMManager *manager = new BMManager(device);
83      BMEffect *effect = manager->CreateEffect(EFFECT_NAME);
        // メイン メッセージ ループ:
86      while(1) {
87          if(PeekMessage(&msg, NULL, 0, 0, PM_NOREMOVE)) {
88              if(!GetMessage(&msg,NULL,0,0)) break;
89              TranslateMessage(&msg);
90              DispatchMessage(&msg);
91          }
92          else {
93              // 更新処理
94              effect->Update();
                if(effect->IsExpired()) {
                    effect->Reset();
97              }
98              // 描画処理
99              device->Clear(0, NULL, D3DCLEAR_TARGET|D3DCLEAR_ZBUF
100             device->BeginScene();
101             manager->Begin();
102             manager->DrawEffect(effect);
                manager->End();
104             device->EndScene();
105             device->Present(NULL, NULL, NULL, NULL);
106         }
107     }
```

MATCHLOCK

3-3：サンプルの構成について

■ クラスの継承関係は、どうなっているの？



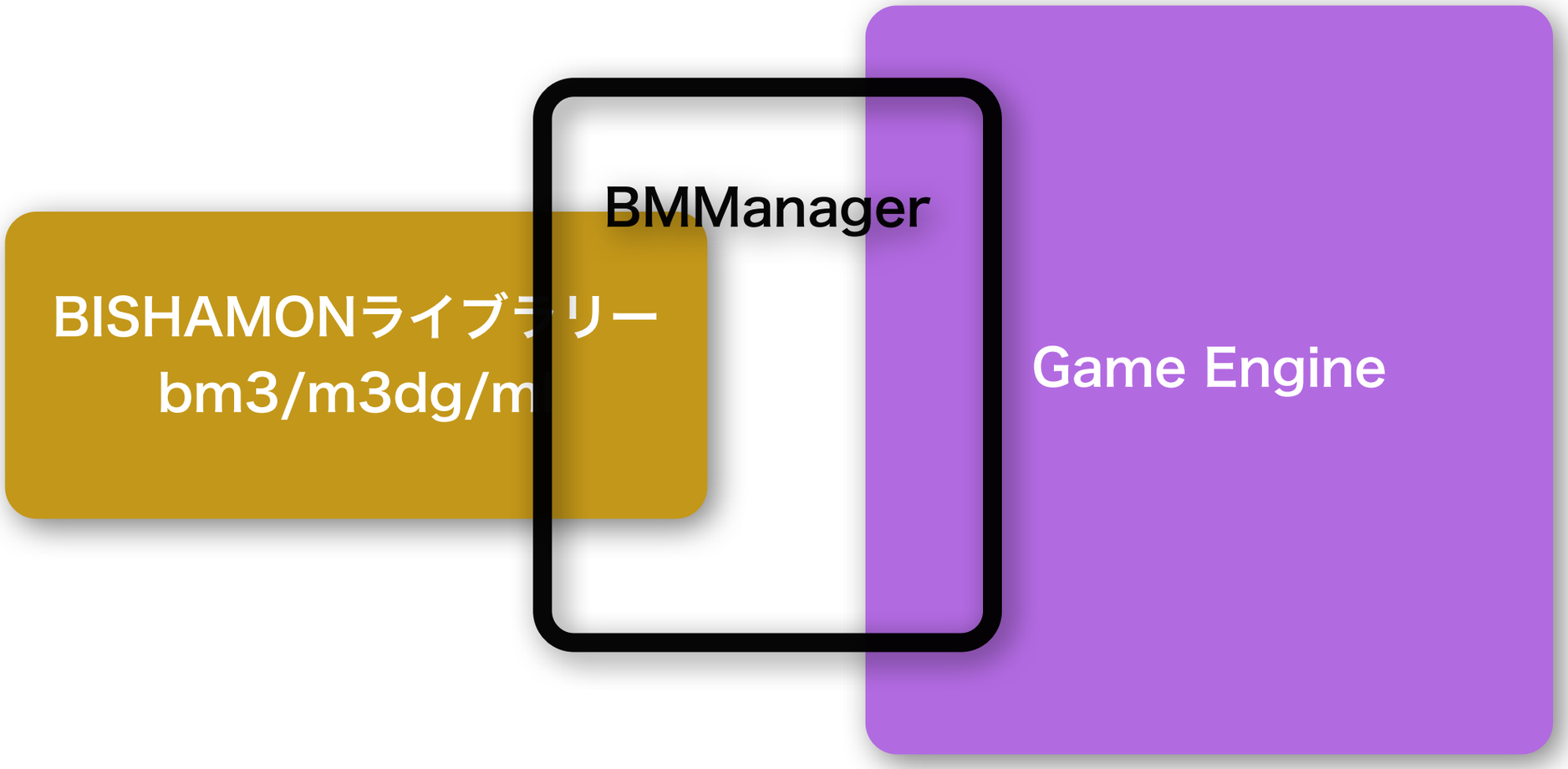
BMManagerが各リソースのコンテナを持っている関係

MATCHLOCK

3-3：サンプルの構成について

■ BMManagerの役割って何？

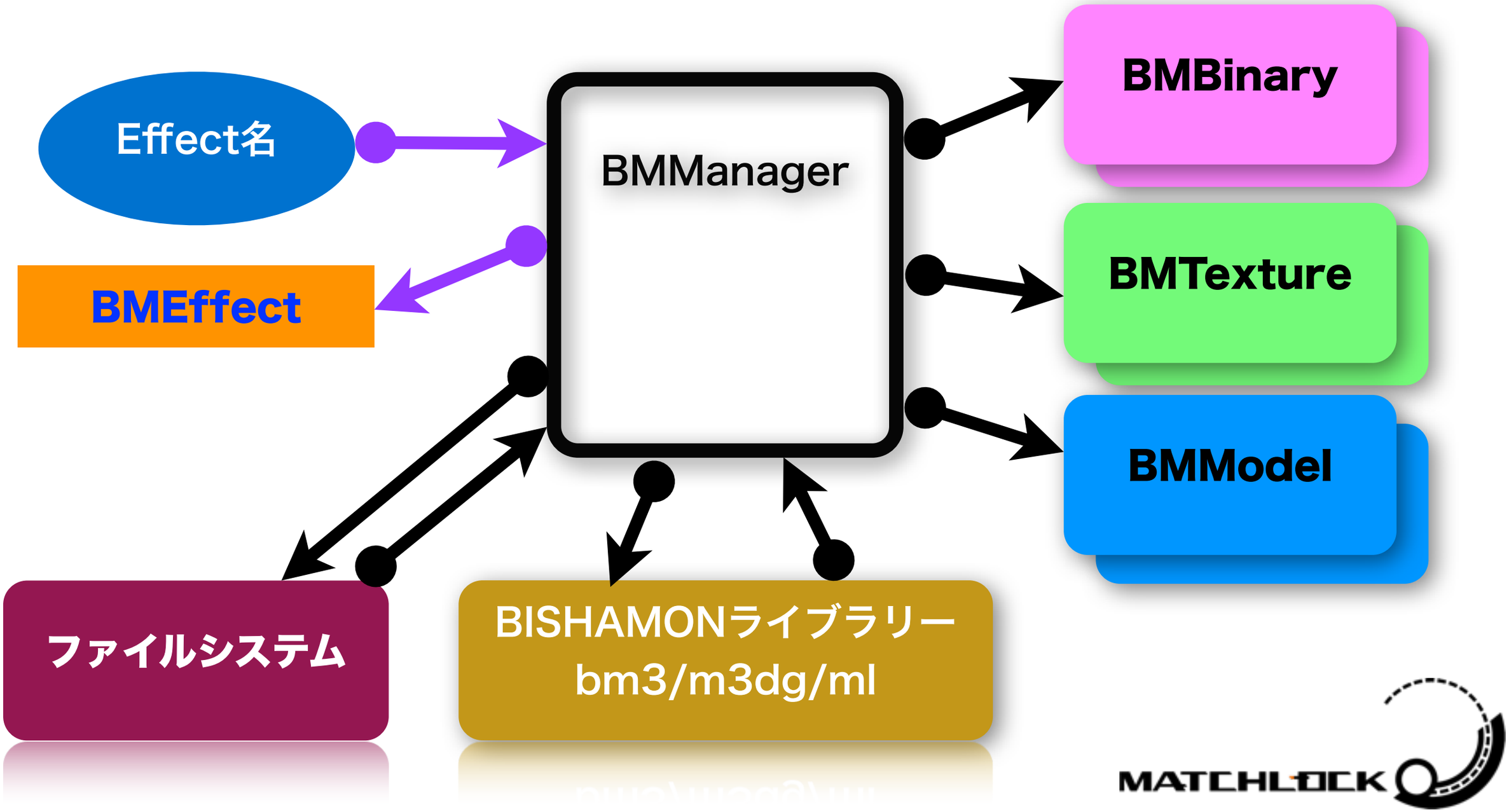
BMManager、および、BM〜のクラスはBISHAMONのライブラリーを利用するため、Personal版用に提供されたサンプルソースです。



3-3：サンプルの構成について

■ BMManagerの役割って何？

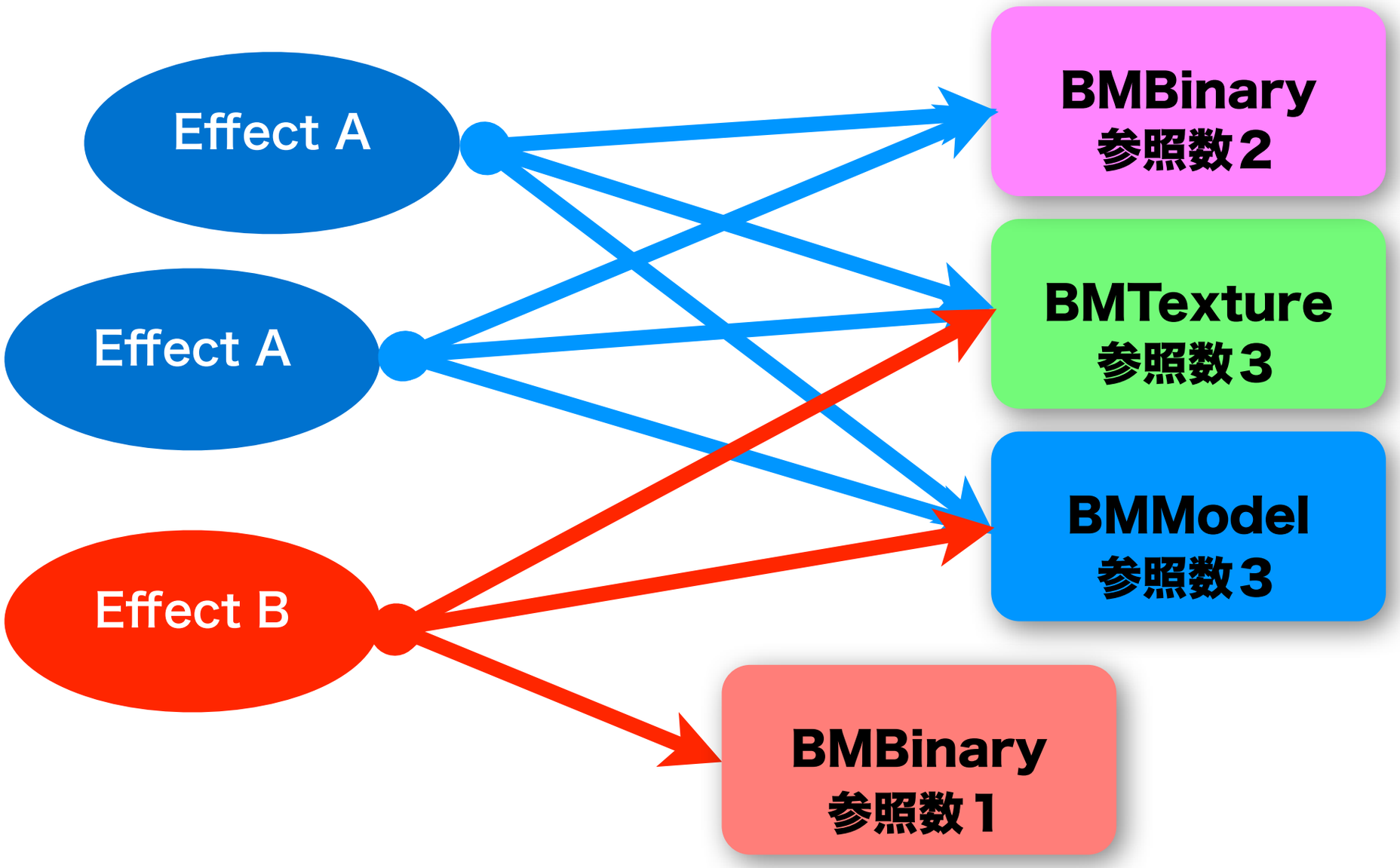
BMManagerの主な目的は、読み込んだBM Effectのリソース(BMBバイナリー、テクスチャー、モデル)を管理しています。



3-3：サンプルの構成について

■ BMManagerの役割って何？

参照カウンタを用いて、複数同じエフェクトや、別のエフェクト同士で、同じリソースを共有し使いまわせるようにしています。



3-3：サンプルの構成について

■ 具体的にはどうするの？

１つエフェクトの注目して、読込＆生成ー＞更新ー＞描画までを説明します。

manager が BMManager のインスタントとします。

BISHAMON の初期化
BM EFFECT の読込＆生成

```
BMEffect* effect =
        manager->CreateEffect(“エフェクトの名前”)
```

BM EFFECT の更新

```
effect->SetMatrix( matrix );  <- エフェクトの向きや位置を更新
effect->Update();
```

BM EFFECT の描画
（カメラ情報が必要）

```
manager->Begin();
manager->DrawEffect( effect );
manager->End();
```

１つのエフェクトに注目すると、これらが最低限必要な処理になります。

MATCHLOCK

3-3：サンプルの構成について

■ 具体的にはどうするの？

全体的は、以下の『カメラ情報を設定』も必要になります。

BISHAMON の初期化
BM EFFECT の読込＆生成

```
BMEffect* effect =
        manager->CreateEffect(“エフェクトの名前”)
```

BM EFFECT の更新

```
effect->SetMatrix( matrix );  <- 向きや位置の更新
effect->Update();
```

BM EFFECT の描画
（カメラ情報が必要）

```
// カメラ情報を設定（ビルボード等の処理のため）
manager->SetView( カメラViewマトリクス );
manager->SetProjection( カメラProjectionマトリクス );

// エフェクト全体を描画
manager->Begin();
manager->DrawEffect( effect );
manager->DrawEffect( 別のエフェクト );
        :     （全エフェクトをDrawEffectする）
manager->End();
```

MATCHLOCK

3-3：サンプルの構成について

■ 結局、BMManager＆BM～は必須なの？

必須ではありません！！

これらは、あくまでも『サンプル』としての提供になります。

実際のゲーム開発では、

・共通エフェクト、キャラクター、ステージ単位でパッケージしたり

・シーングラフで管理したり

など、それぞれのゲーム・エンジンに適した『リソース管理システム』に組み込むのがベストです。

さて、次はいよいよ実際のゲームを使った『導入』について解説をします！！

MATCHLOCK

第３部
BISHAMONの導入と実践

- 3-1：インストールしてみる
- 3-2：サンプルを動かしてみる
- 3-3：サンプルの構成について
- **3-4：STGに導入してみる**

MATCHLOCK

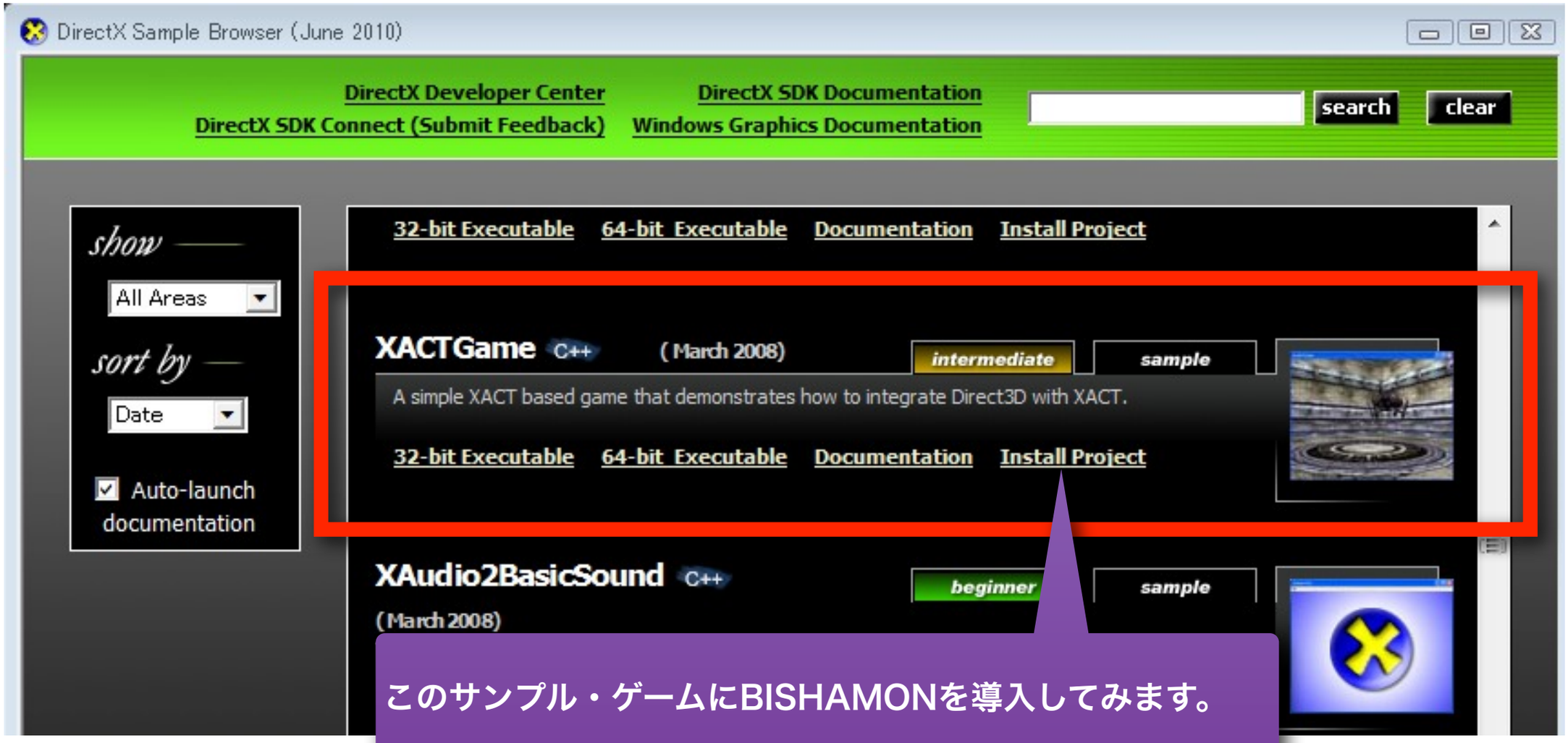
3-4：STGに導入してみる

■ DirectXのサンプルにあるゲームを使って、
BISHAMONを導入してみます

「スタートメニュー」から、
ー＞　すべてのプログラム
ー＞　Microsoft DirectX SDK (June 2010)
ー＞『DirectX Sample Browser』
を選択し、起動します。

3-4：STGに導入してみる

■ DirectXのサンプルにあるゲームを使って、
BISHAMONを導入してみます

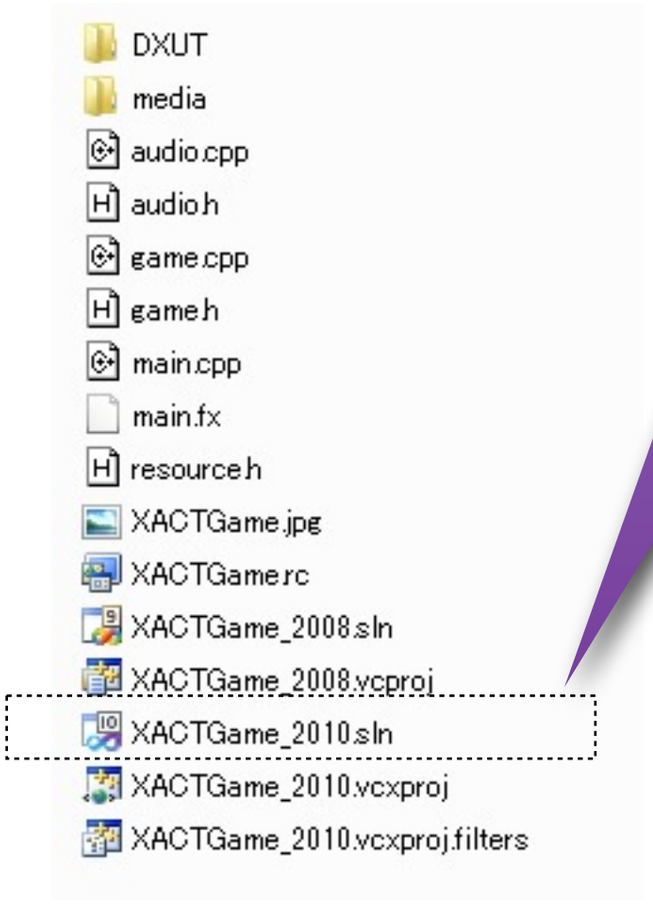


MATCHLOCK

3-4：STGに導入してみる

■ XACTGameサンプルってどんなゲーム？

XACTGame



ダブル・クリックして、VS2010を起動し、実行

3-4：STGに導入してみる

■ XACTGameサンプルってどんなゲーム？

MATCHLOCK

3-4：STGに導入してみる

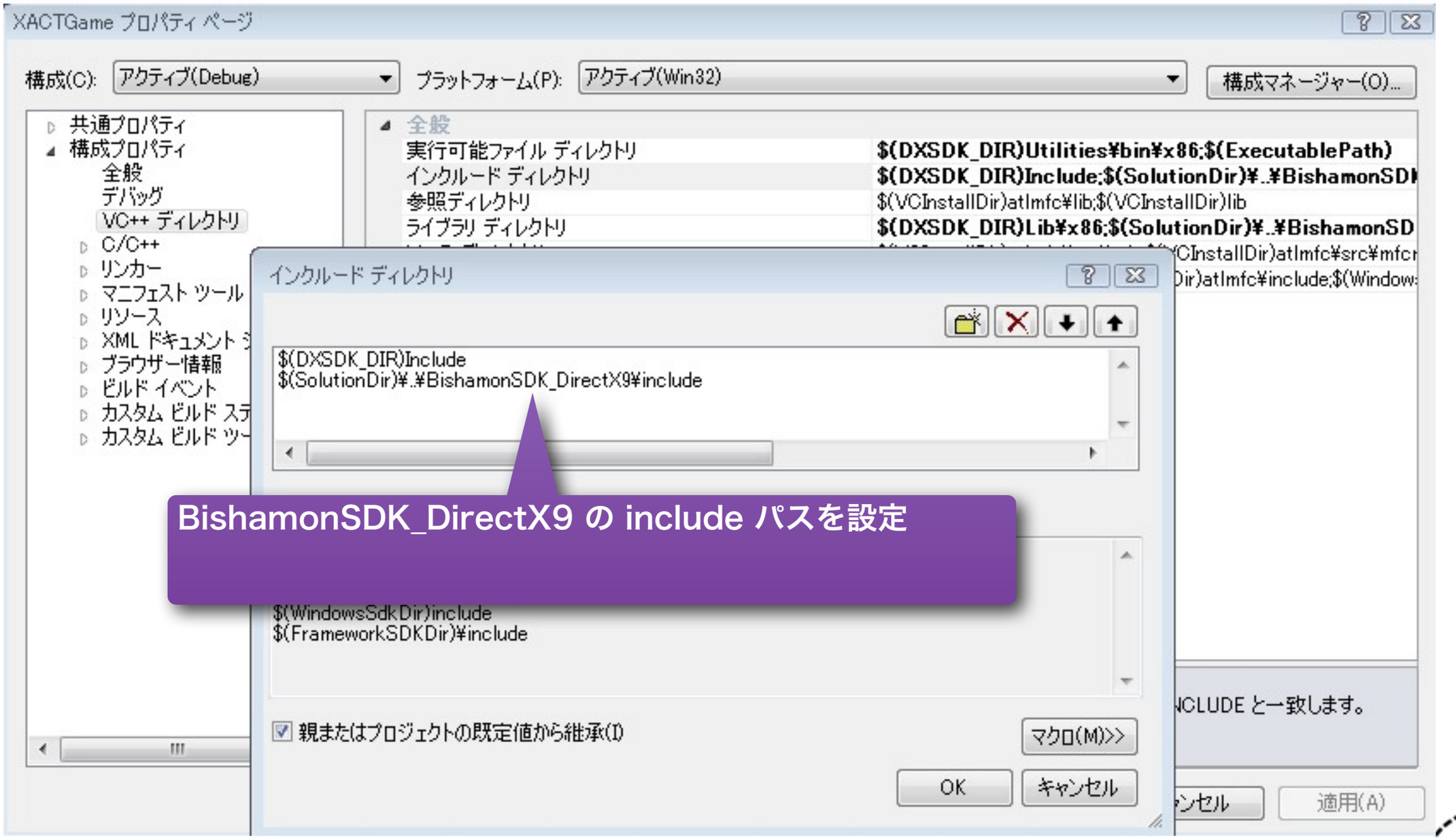
■ BISHAMON SDK とのフォルダ構成

BishamonSDK_DirectX9 の include, lib/vc10 へのパスを設定するため、右のようにフォルダ構成にしました。

「simple.sln」のときと同じように、
- XACTGameのプロジェクトを右クリック
- 「プロパティ」を選択してください。

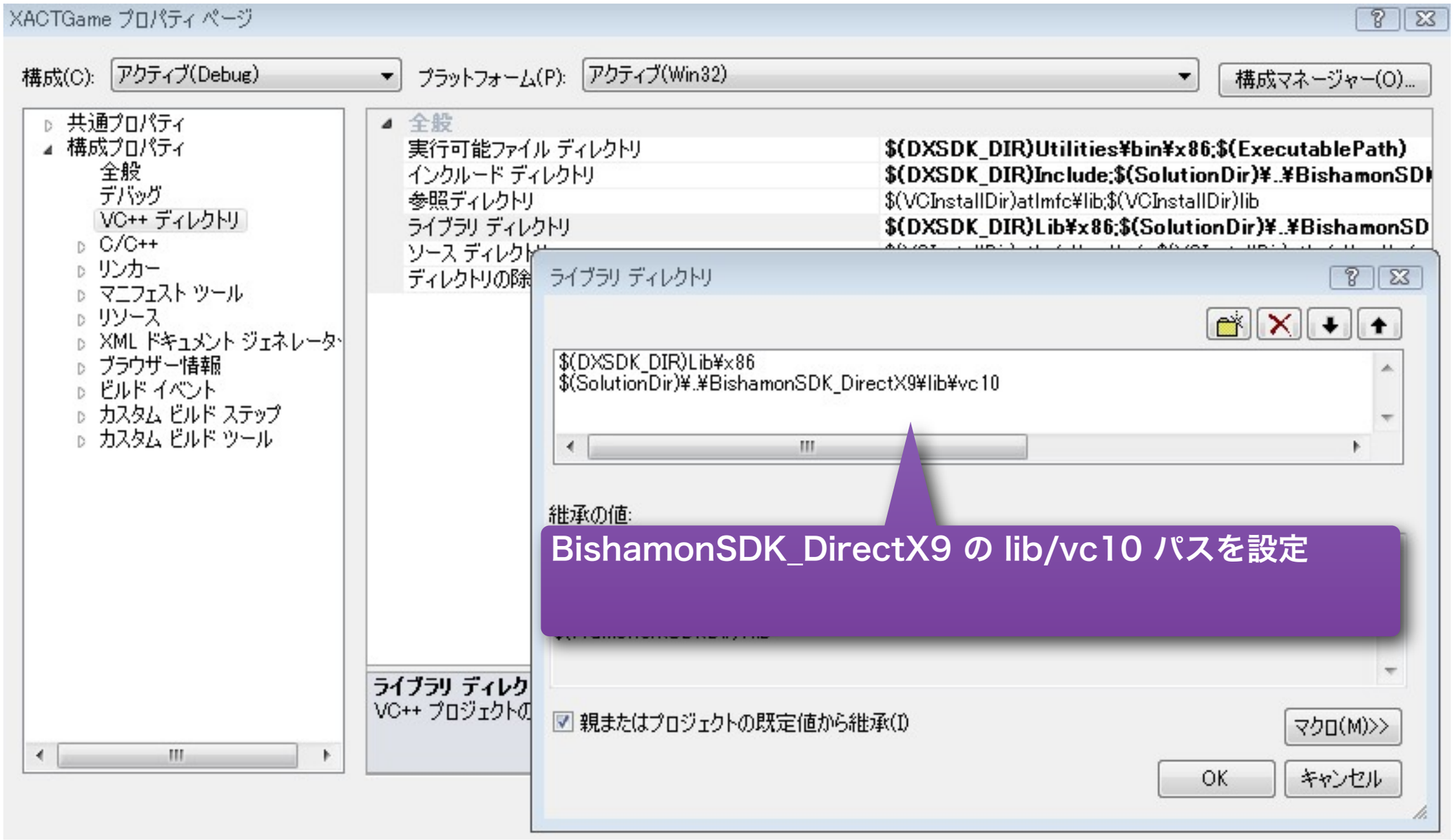
3-4：STGに導入してみる

■ BISHAMON SDK へのパスを設定



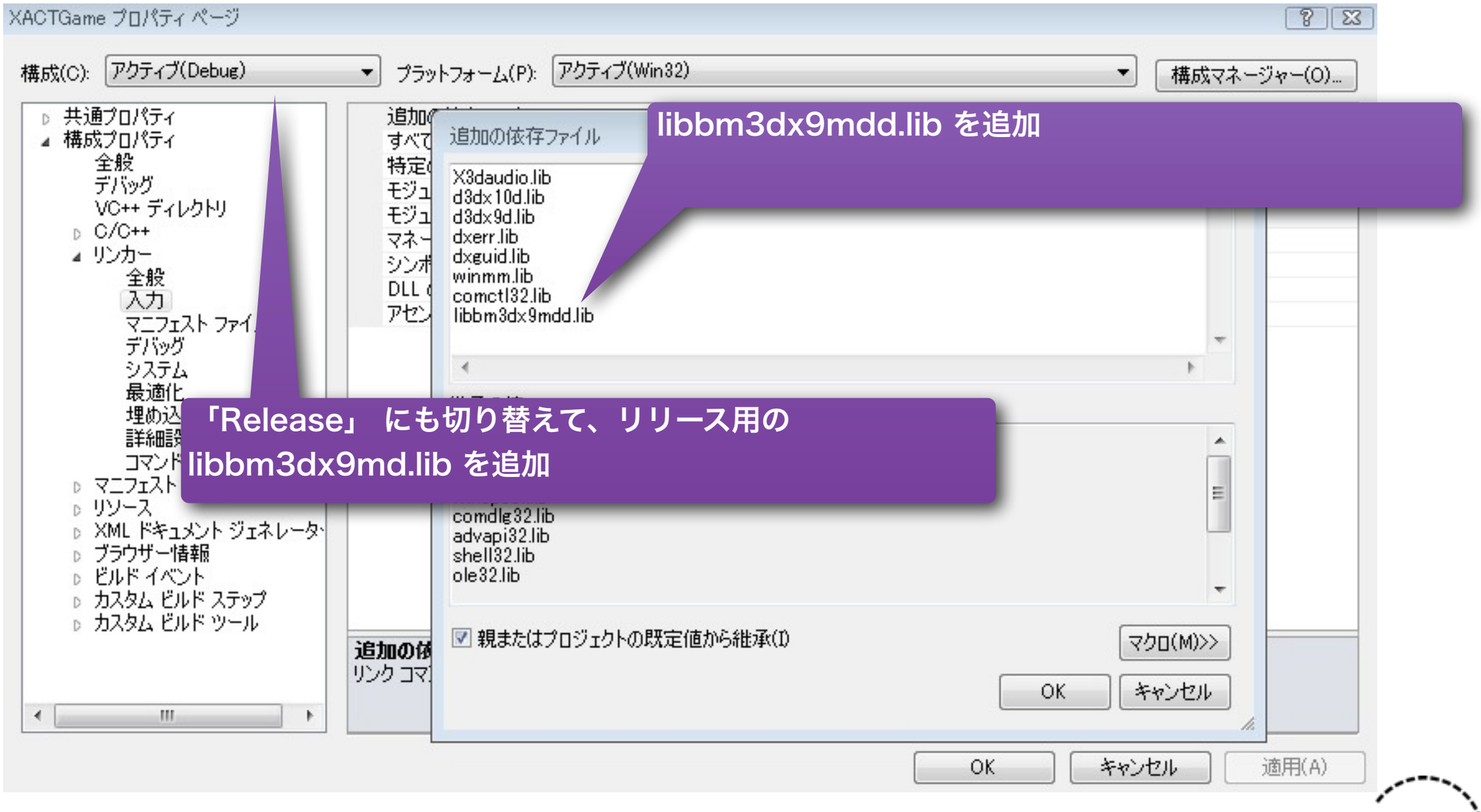
3-4：STGに導入してみる

■ BISHAMON SDK へのパスを設定



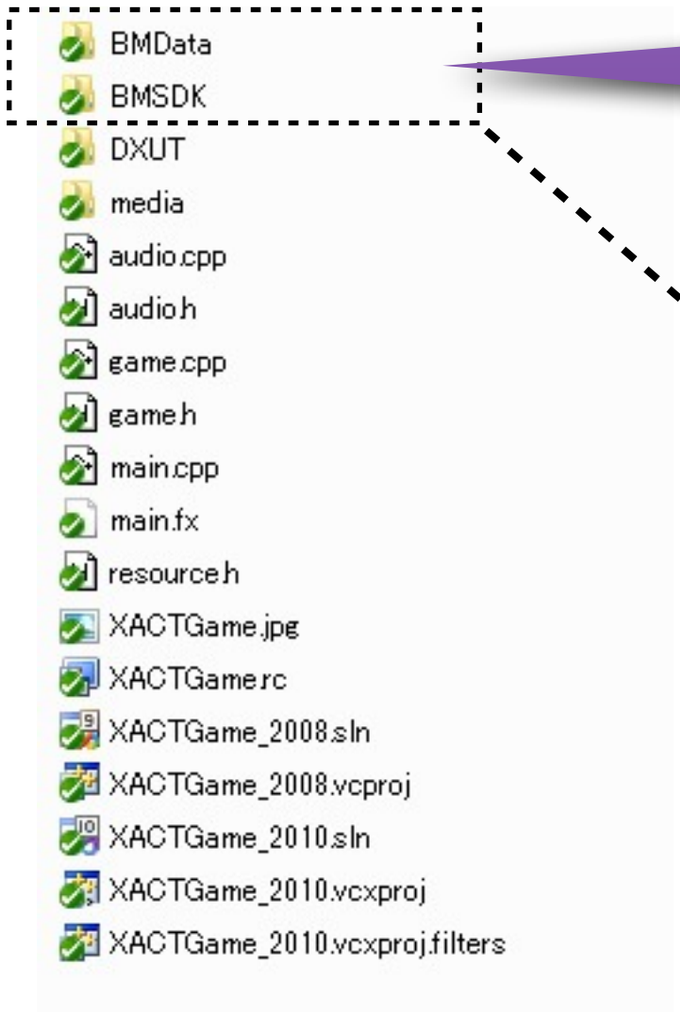
3-4：STGに導入してみる

■ BISHAMON SDK へのパスを設定



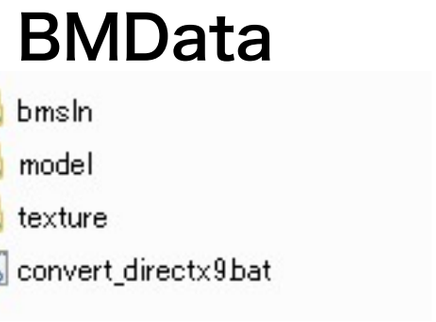
3-4：STGに導入してみる

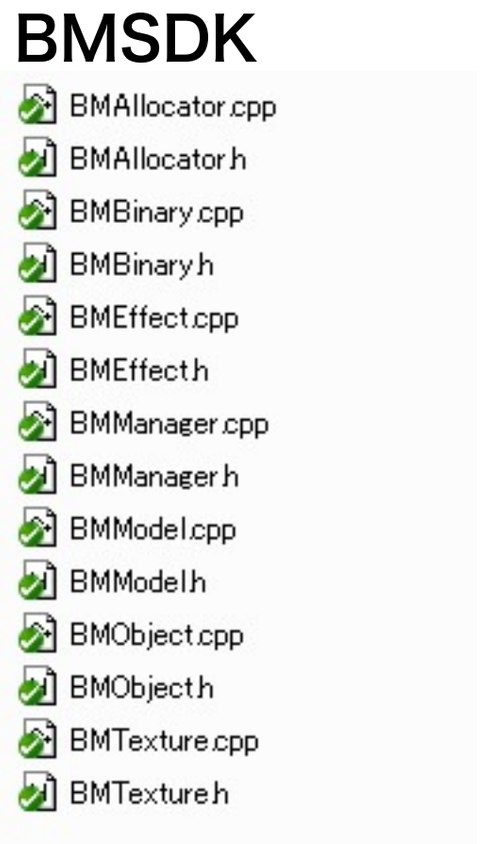
■ BM～のファイルを追加



BMDataフォルダ追加

・BMData .... BISHAMON のデータ用フォルダ
・BMSDK　.... BM～ファイル用フォルダ

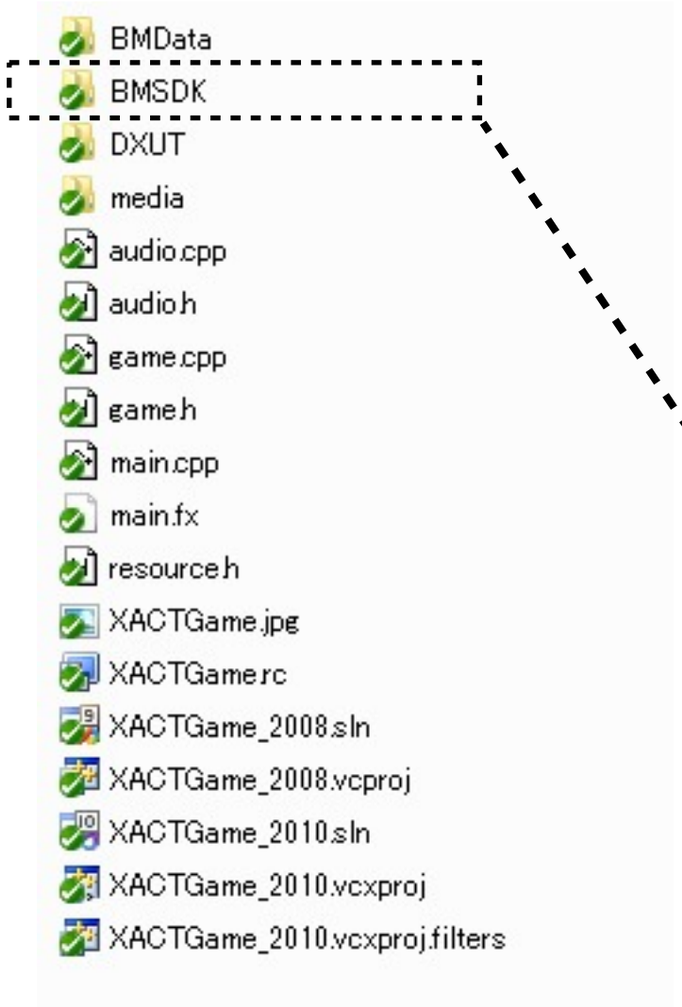


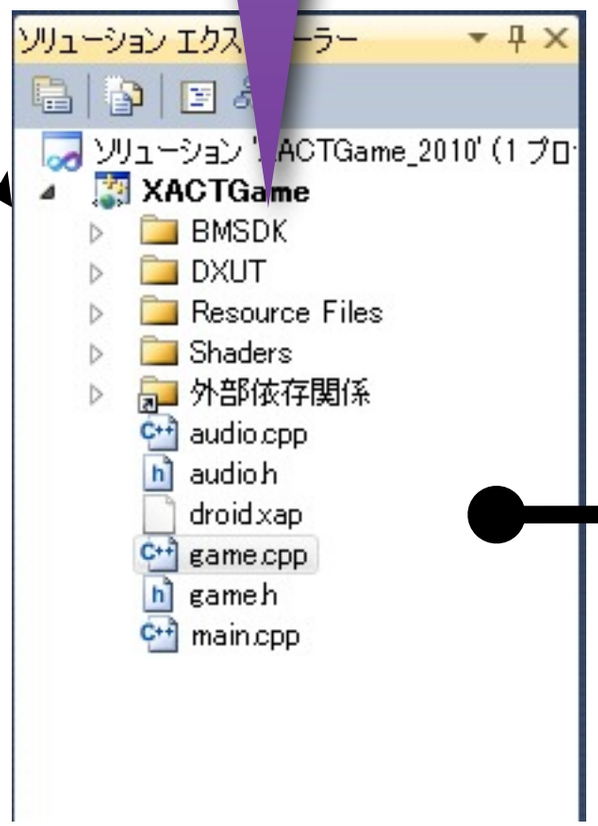


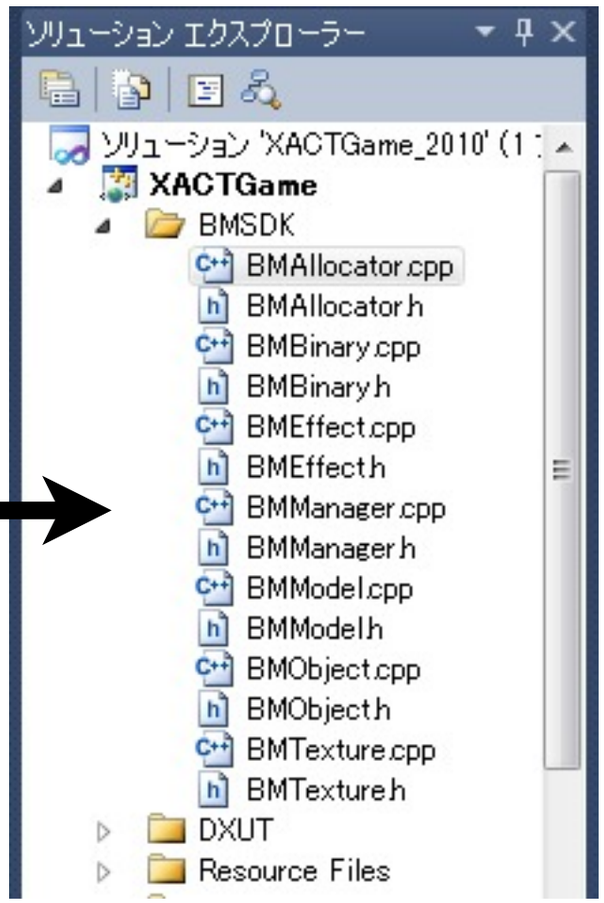
3-4：STGに導入してみる

■ BM~のファイルをプロジェクトへ追加

XACTGameプロジェクトに、
・「BMSDK」フォルダを生成し、
・BMSDK内のBM~ファイルを追加







3-4：STGに導入してみる

■ BMManager をちょっと使いやすいように修正

- 全BM〜ファイルの「stdafx.h」 -> 「DXUT.h」 へ変更
（プリコンパイルヘッダーを合わせる）
- データベースフォルダのパスを設定できるように修正
- BMManager をデフォルトコンストラクタをpublic へ
- Create() をメソッド追加
- BMEffect の管理用リストを追加
- UpdateAllEffect() / DrawAllEffect() を追加
- DestroyAllEffect() / Destroy() を追加
- BMEffectに、SetLoop() / IsLoop() を追加
- 上記修正に影響のある箇所にいくつか修正

詳しくは、ソースコードを参照してください！！

MATCHLOCK

3-4：STGに導入してみる

■ game.cpp に、『導入』

- グローバル変数に追加
- 初期化に追加
- 更新処理に追加
- 描画処理に追加
- 終了処理に追加
- 死亡時にエフェクト追加
- 壁バウンドにエフェクト追加
- ロストデバイスに追加
- リセットデバイスに追加

追加した箇所には、
// BISHAMON
のコメントをつけています。
詳しくは、ソースコードを確認してください。

MATCHLOCK

3-4：STGに導入してみる

■ game.cpp に、『導入』

- グローバル変数に追加
- 初期化に追加
- 更新処理に追加
- 描画処理に追加
- 終了処理に追加

バイナリーの読込のみ先にできるように、
管理リストを用意

```
 1 //--------------------------------------------------------
 2 // File: game.cpp
 3 //
 4 // Copyright (c) Microsoft Corporation. All rights reserve
 5 //--------------------------------------------------------
 6 #include "DXUT.h"
 7 #include "SDKmisc.h"
 8 #include "BMSDK/BMManager.h"
 9 #include "BMSDK/BMEffect.h"
10 #include "BMSDK/BMBinary.h"
11 #include <stdio.h>
12 #include "game.h"

53 //--------------------------------------------------------
54 // Global variables
55 //--------------------------------------------------------
56 CFirstPersonCamera  g_Camera;
57 RENDER_STATE        g_Render;
58 GAME_STATE          g_GameState;
59
60 // BISHAMON BMManager
61 BMManager           g_BMManager;
62
63 // BISHAMON バイナリー先行読み込み用
64 typedef std::vector<const wchar_t*> BMBinaryListType;
65 BMBinaryListType    g_BMBinaryList;
66
67 // BISHAMON CONST
68 const wchar_t* BM_DATABASE_DIR = L"./BMData/";
69 const wchar_t* BMEFFECT_BOMB = L"hit09_bomb_ks02";
70 const wchar_t* BMEFFECT_HIT = L"gem_always_ks01";
71
72
73 //--------------------------------------------------------
```

MATCHLOCK

3-4：STGに導入してみる

■ game.cpp に、『導入』

- グローバル変数に追加
- 初期化に追加
- 更新処理に追加
- 描画処理に追加
- 終了処理に追加

OnCreateDevice

```
466 ····//·Rendering·technique·(FF·or·shader)↵
467 ····SetEffectTechnique();↵
468 ····g_Render.pEffect->SetBool(·"g_bUseSpecular",·!g_Rende
469 ····g_Render.pEffect->SetBool(·"g_bUseAnisotropic",·g_Ren
470 ····g_Render.pEffect->SetInt(·"g_MinFilter",·g_Render.Use
471 ····g_Render.pEffect->SetInt(·"g_MaxAnisotropy",·g_Render
472 ····g_Render.pEffect->SetTexture(·g_Render.hNormalMap,·g_
473 ↵
474 ····→//·BISHAMON·Initialize↵
475 ····→g_BMManager.Create(pd3dDevice,·BM_DATABASE_DIR/*BM·Da
476 ↵
477 ····→//·BISHAMON·BMBを先行で読み込んでおく↵
478 ····→g_BMBinaryList.push_back(·BMEFFECT_BOMB·);↵
479 ····→g_BMBinaryList.push_back(·BMEFFECT_HIT·);↵
480 ····→BMBinaryListType::iterator·ite·=·g_BMBinaryList.begin
481 ····→BMBinaryListType::iterator·end·=·g_BMBinaryList.end()
482 ····→for·(·;·ite·!=·end·;·++ite·){↵
483 ····→····→g_BMManager.LoadBinary(·*ite·);↵
484 ····→}↵
485 ↵
486 ····→return·S_OK;↵
487 }↵
```

MATCHLOCK

3-4：STGに導入してみる

■ game.cpp に、『導入』

グローバル変数に追加

初期化に追加

更新処理に追加

描画処理に追加

終了処理に追加

OnFrameMove

```
1400
1401    HandleAmmoAI( fElapsedTime );
1402    HandleDroidAI( fElapsedTime );
1403
1404    // BISHAMON Update
1405    g_BMManager.UpdateAllEffect();
1406 }
1407
```

OnFrameRender

```
1662                    RenderDroid( pd3dDevice, A, mView, mProj
1663                }
1664            }
1665        }
1666
1667        // BISHAMON Render Draw
1668        g_BMManager.DrawAllEffect(g_Camera.GetViewMatrix(),
1669
1670        // Render transparent objects last
1671        if( g_GameState.nAmmoCount > 0 )
1672        {
1673            for( int A = 0; A < MAX_AMMO; A++ )
1674            {
```

MATCHLOCK

3-4：STGに導入してみる

■ game.cpp に、『導入』

グローバル変数に追加

初期化に追加

更新処理に追加

描画処理に追加

終了処理に追加

OnDestroyDevice

```
2126 void CALLBACK OnDestroyDevice( void* pUserContext )
2127 {
2128     // BISHAMON 先行で読み込んだバイナリーの破棄
2129     BMBinaryListType::iterator ite = g_BMBinaryList.begin()
2130     BMBinaryListType::iterator end = g_BMBinaryList.end();
2131     for ( ; ite != end ; ++ite ){
2132         g_BMManager.ReleaseBinary( *ite );
2133     }
2134     g_BMBinaryList.clear();
2135
2136     // BISHAMON Destroy
2137     g_BMManager.Destroy();
2138
2139     g_Render.DialogResourceManager.OnD3D9DestroyDevice();
2140     SAFE_RELEASE( g_Render.pEffect );
2141     SAFE_RELEASE( g_Render.pFont );
2142     SAFE_RELEASE( g_Render.pDefaultTex );
2143     SAFE_RELEASE( g_Render.pDefaultNormalMap );
2144     SAFE_RELEASE( g_Render.pDroidNormalMap );
```

MATCHLOCK

3-4：STGに導入してみる

■ game.cpp に、『導入』

死亡時にエフェクト追加

壁バウンドにエフェクト追加

ロストデバイスに追加

リセットデバイスに追加

CheckForAmmoToDroidCollision

```
979 ·····//·Check·if·the·B·instances·are·already·moving·aw
980 ·····//·If·so,·skip·collision.··This·can·happen·when·a
981 ·····//·bunched·up·next·to·each·other.↲
982 ·····float·fImpact·=·D3DXVec3Dot(·&vAToB,·&g_GameState
983 ·····D3DXVec3Dot(·&vAToB,·&g_GameState.DroidQ[B].v
984 ·····if(·fImpact·>·0.0f·)↲
985 ·····{↲
986 ·····g_GameState.AmmoQ[A].bActive·=·false;↲
987 ·····g_GameState.DroidQ[B].vNudgeVelocity·+=·g_Gam
988 ↲
989 ·····g_GameState.DroidQ[B].nHitPoints--;↲
990 ·····if(·g_GameState.DroidQ[B].nHitPoints·<=·0·)↲
991 ·····{↲
992 ·····→·→·→·→·→const·D3DXVECTOR3&·pos·=·g_GameState.Ammo
993 ·····→·→·→·→·→ml::matrix44·matrix;↲
994 ·····→·→·→·→·→matrix.set_unit();↲
995 ·····→·→·→·→·→matrix.set_scale(0.1,0.1,0.1);↲
996 ·····→·→·→·→·→matrix.set_translate(pos.x,·pos.y,·pos.z)
997 ↲
998 ·····→·→·→·→·→//·BISHAMON·死亡エフェクト↲
999 ·····→·→·→·→·→BMEffect*·effect·=·g_BMManager.CreateEffe
1000 ·····→·→·→·→·→if·(effect·!=·NULL){↲
1001 ·····→·→·→·→·→·→effect->SetMatrix(·matrix·);↲
1002 ·····→·→·→·→·→}↲
1003 ↲
1004 ·····g_GameState.DroidQ[B].fDeathAnimation·=·0
1005 ·····Play3DAudioCue(·g_audioState.iDroidDestro
```

MATCHLOCK

3-4：STGに導入してみる

■ game.cpp に、『導入』

死亡時にエフェクト追加

壁バウンドにエフェクト追加

ロストデバイスに追加

リセットデバイスに追加

```
1078 //------------------------------------------------------------------------
1079 void CreateBoundEffect(const D3DXVECTOR3& pos)
1080 {
1081     // BISHAMON ヒット・エフェクト
1082     ml::matrix44 matrix;
1083     matrix.set_unit();
1084     matrix.set_scale(0.1, 0.1, 0.1);  //<- 1/10 倍にする
1085     matrix.set_translate(pos.x, pos.y, pos.z);
1086
1087     BMEffect* effect = g_BMManager.CreateEffect(BMEFFECT_HIT);
1088     if (effect != NULL) {
1089         effect->SetMatrix( matrix );
1090     }
1091 }
1092
1093 void CheckForAmmoToWallCollision( int A )
1094 {
```

CheckForAmmoToWallCollision

```
1143         g_GameState.AmmoQ[A].vPosition.z = g_MinBound.z + ( AM
1144         g_GameState.AmmoQ[A].vVelocity.z = -g_GameState.AmmoQ[
1145         PlayAudioCue( g_audioState.iAmmoBounce );
1146             // BISHAMON EFFECT
1147             CreateBoundEffect(g_GameState.AmmoQ[A].vPosition);
1148     }
1149     if( g_GameState.AmmoQ[A].vPosition.z > g_MaxBound.z - ( AM
1150     {
1151         g_GameState.AmmoQ[A].vPosition.z = g_MaxBound.z - ( AM
1152         g_GameState.AmmoQ[A].vVelocity.z = -g_GameState.AmmoQ[
1153         PlayAudioCue( g_audioState.iAmmoBounce );
1154             // BISHAMON EFFECT
1155             CreateBoundEffect(g_GameState.AmmoQ[A].vPosition);
1156     }
1157
1158     // Check bounce on left and right walls
1159     if( g_GameState.AmmoQ[A].vPosition.x < g_MinBound.x + ( AM
1160     {
1161         g_GameState.AmmoQ[A].vPosition.x = g_MinBound.x + ( AM
1162         g_GameState.AmmoQ[A].vVelocity.x = -g_GameState.AmmoQ[
1163         PlayAudioCue( g_audioState.iAmmoBounce );
1164             // BISHAMON EFFECT
1165             CreateBoundEffect(g_GameState.AmmoQ[A].vPosition);
1166     }
1167     if( g_GameState.AmmoQ[A].vPosition.x > g_MaxBound.x - ( AM
1168     {
1169         g_GameState.AmmoQ[A].vPosition.x = g_MaxBound.x - ( AM
1170         g_GameState.AmmoQ[A].vVelocity.x = -g_GameState.AmmoQ[
1171         PlayAudioCue( g_audioState.iAmmoBounce );
1172             // BISHAMON EFFECT
1173             CreateBoundEffect(g_GameState.AmmoQ[A].vPosition);
1174     }
1175 }
1176
```

MATCHLOCK

3-4：STGに導入してみる

■ game.cpp に、『導入』

- 死亡時にエフェクト追加
- 壁バウンドにエフェクト追加
- ロストデバイスに追加
- リセットデバイスに追加

OnLostDevice

```
2100 void CALLBACK OnLostDevice( void* pUserContext )
2101 {
2102     // BISHAMON Lost Device
2103     g_BMManager.DeviceLost();
2104
2105     g_Render.DialogResourceManager.OnD3D9LostDevice();
2106     if( g_Render.pFont )
2107         g_Render.pFont->OnLostDevice();
2108     if( g_Render.pEffect )
2109         g_Render.pEffect->OnLostDevice();
2110     SAFE_RELEASE( g_Render.pTextSprite );
2111     g_Render.meshCell.InvalidateDeviceObjects();
```

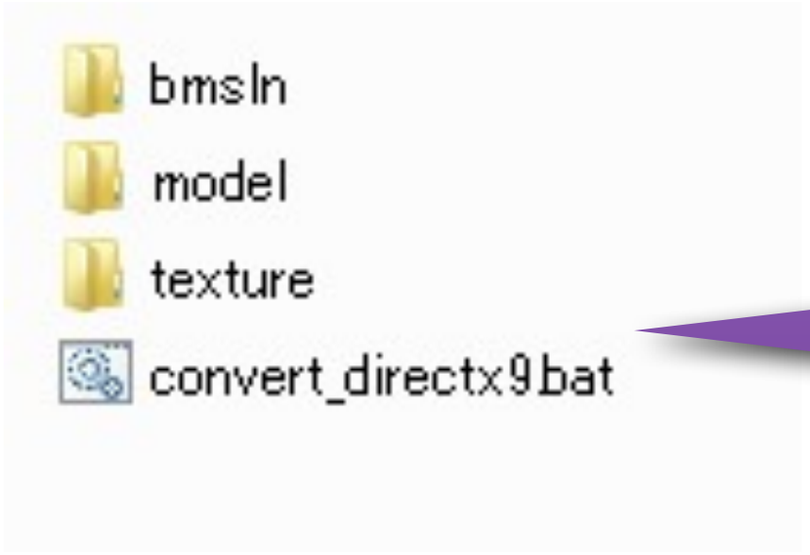
OnResetDevice

```
605 HRESULT CALLBACK OnResetDevice( IDirect3DDevice9* pd3dDevice
606                                 const D3DSURFACE_DESC* pBack
607 {
608     HRESULT hr;
609
610     // BISHAMON ResetDevice
611     g_BMManager.ResetDevice( pd3dDevice );
612
613     V_RETURN( g_Render.DialogResourceManager.OnD3D9ResetDevi
614
615     if( g_Render.pFont )
616         V_RETURN( g_Render.pFont->OnResetDevice() );
```

MATCHLOCK

3-4：STGに導入してみる

■ バイナリーコンバート用バッチファイル修正



細かいことですが、このコンバートバッチの中のEXEへのパスもフォルダ構成に合わせて修正

あとは、ここのフォルダにBISHAMON のデータを入れれば、コンバートして利用できるようになります！

MATCHLOCK

3-4：STGに導入してみる

■ 以上で、導入完了！！

MATCHLOCK

ゲームエフェクトセミナー

第３部：総まとめ

- 3-1：インストールしてみる
  （必要なもの、フォルダ構成）
- 3-2：サンプルを動かしてみる
  （simpleサンプルについて）
- 3-3：サンプルの構成について
  （BMManagerの役割と構成）
- 3-4：STGに導入してみる
  （導入する箇所について）

MATCHLOCK

ゲームエフェクトセミナー

第３部：総まとめ

使用したソースコード

・Bishamon Personal 体験版
・XACTGame 用に修正した修正分のソースコード

は、下記よりダウンロードできます。

http://www.matchlock.co.jp/products/

注意）
- BishamonSDK for DirectX9
- XACTGame
のソースは正規の場所からダウンロードし、修正分のソースコードを上書きしてください。

MATCHLOCK

ゲームエフェクトセミナー

第３部

BISHAMONの導入と実践

お疲れ様でした

MATCHLOCK