９．バックアップ・アンドゥ

９－１．概要

多くの処理において、バックアップ・アンドゥの機構を用意した。ユーザーは一つの編集ダイアログを開き、ある操作を行う。その結果が期待しているものとは異なる場合に、データや環境設定を元に戻し、あるいは条件を変えて再び操作を行う。このようなバックアップ・アンドゥの機能が用意されていていなければ、ユーザーは個々の操作に際して、必ず成功しなければならない、というプレッシャーに晒される。実物としての模型製作や実際の建設とは異なって、コンピュータ上のデータとしてシミュレーションを行うことの効用の一つは、やり直しのコストが低いという点にあることは間違いない。

しかし、バックアップ・アンドゥの処理ためには、メモリやファイル等のリソースを必要とする上、バックアップ・アンドゥ系自体が信頼性の低いものである場合（例えばバックアップやアンドゥの処理中にシステムダウンを生じる最悪の場合等)、その存在意義には疑問符が付くこととなる。

これまでに試行され、実装されたバックアップ・アンドゥには、Ver.2.09 で不採用とした方法を含め、以下のものがある。

（1）ヒストリー方式

景観シミュレータの開発初期においては、一括してバックアップ・アンドゥを処理するために、ヒストリー機能が構想され、一定レベルまで開発を進めた。これは、ユーザーの操作をコード化し、一連の操作の記録として小さな外部テキストファイルに保存するという機能である。編集操作中にシステムダウンが生じた場合には、少し時間はかかるが、ヒストリーとして記録されているデータを忠実に再現すれば、システムダウンを生じる直前までの編集操作を再現することができる。

但し、この方法には以下の２点で難点があったため、現在では基幹部分のビルドから外している。

①新たな機能を追加する度に、そのコーディング規則を定め、各処理ダイアログに組み込むと共に、外部の記録ファイルのエンコード・デコード機能を拡充しなければならない。

②ヒストリー機能自体に内在するバグが、システム障害の原因となる場合がある。

理由②は、基本的なライブラリ関数にもまだバグが多く残されていた時期であったことによる。現時点で、ヒストリー機能を再検討すると共に、通信的環境におけるコラボレーションのための同期的・協調的な動作など、新たな使用方法に向けた可能性がある。

編集操作に際しては、ヒストリー機能により、保存する意味のある操作自体がコーディングされ、ヒストリーとして累積的に記録される。ある処理を行っている最終にシステムダウンが生じたような場合には、処理を開始した時点でのシステム状態（あるファイルがロードされた状態）を再現し、そこからヒストリーとして記録された操作を再現することにより、リスクに際限を設けることができる。

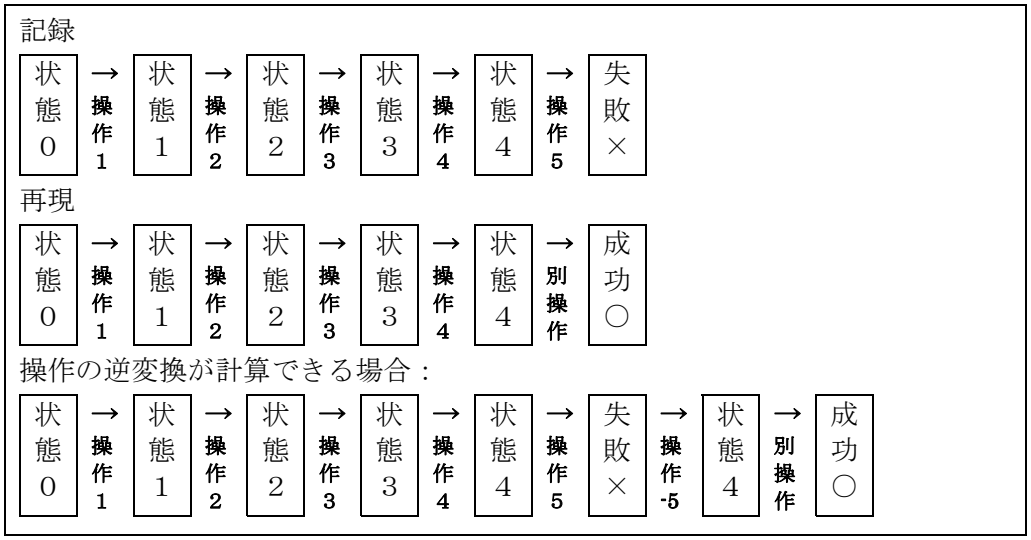

図９－１：ヒストリー方式：ダイアグラム

（２）モデルのファイル保存とアンドゥ

全地物データに大きな改変を加える複雑な処理（地形の自動的な再編集など）に際しては、その前の状態をファイルに保存しておき、結果に不満であれば復元するような仕組みを用意した。

これは、U３ライブラリの機能として実現している。このパターン化した処理においては、バックアップ構造体に、処理前の状態を保存する一時的ファイルの名称を記録する。復元処理においては、この一時的ファイルを復元する。

（３）メモリ上での保存とアンドゥ

バックアップすべき情報の量が、メモリ使用量による制約に影響しない程度に十分小さければ、メモリ上にバックアップ情報を保存しておき、アンドゥに利用することができる。

上記の（２）及び（３）の基本的な処理の流れは下記のようなものとなる。

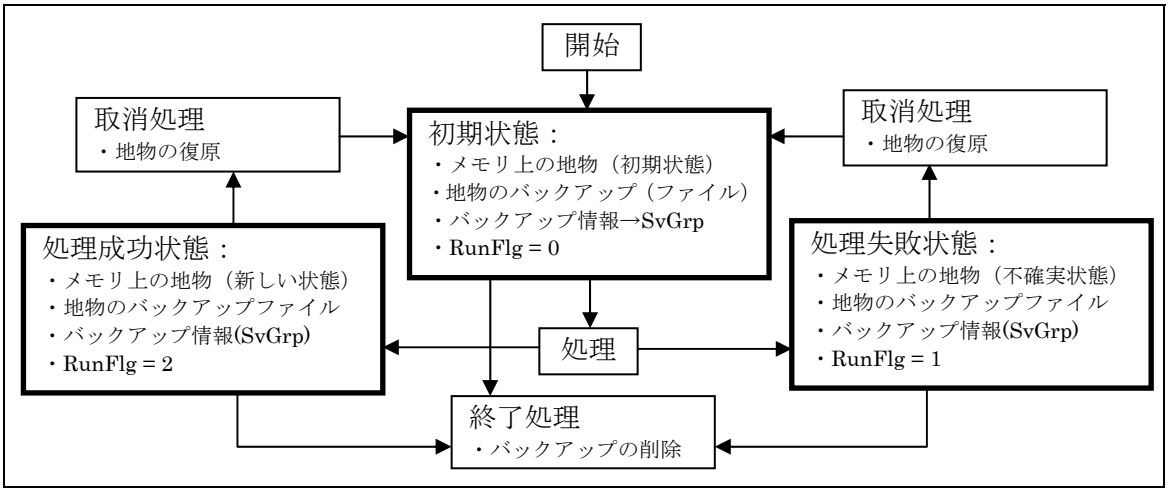

図９－２：地物のバックアップファイルによるアンドゥ処理

９－２．モデル全体のバックアップとアンドゥ

前述（２）で述べたモデルの保存とアンドゥのために、Ｕ３ライブラリの中で関数を用意してある。これは、基幹部分の配置コマンドで用いている方法であるが、同じライブラリ関数をプラグインＤＬＬからも利用可能である。

例えば、園路生成の場合では、RunFlag は、enroOpt.RunFlg として、SvGrp は、enroParam.SvGrp として実装している。このダイアログにおいては、ユーザーが諸条件を設定した後に、地形を大きく改変するような図形演算処理を行っている。このために、編集処理結果（失敗する場合）を見て、やり直すことを想定し、キャンセル・ボタンによって復原・終了するのではなく、復原の後に、再試行することを可能としている。終了処理（出口）は、終了ボタンに統一している。従って、終了ボタンが押されるのは、①何も処理が行われないまま、直ちに終了する場合、②処理が行われ、成功している状態、③処理が行われ、失敗している状態がありうる。

具体的に、以下のように処理している。

① 編集開始時点

CMyOrthoVW::OnPaint() において

giFirst フラグが－１（初回表示）または０（二回目以降、編集機能が起動された）の場合に、編集が開始されたことを認識し、以下のバックアップ処理を実行する。

enroParam.SvGrp = u3SaveGroupFamily(*enroParam.OrgGrp);

これにより、一時的なファイルが作成され、ファイル名等の情報が、SvGrp に記録される。

RunFlg を０にセットする。

② 実行が押された場合

CParkRoadWnd::OnBtnRun()の中で、編集処理が実行される。

RunFlg を調べ、初期状態でない場合には処理を行わない。

enroOpt.RunFlg を１にセットし、実行中であることを示した上で処理を開始する。

処理が正常に終了した場合には、RunFlg を２にセットする。

エラーで終了した場合、地物データは処理前のものとも、正常に処理された場合のものとも異なっている可能性がある。

③ 編集が行われた後、キャンセルが押された場合

1) enroOpt.RunFlg を０にセットする（実行前の状態）。

2) ルート・グループ（オリジナルとは異なる編集されたデータ）を解放する。

3)*enroParam.OrgGrp = u3RestoreGroupFamily(TRUE, enroParam.SvGrp);

この操作で、バックアップファイルは削除され、オリジナルのモデルが復元される。

SvGrp のメモリブロックは解放される。

4) enroParam.SvGrp = u3SaveGroupFamily(*enroParam.OrgGrp);

バックアップファイルが再び作成され、新たに構築された SvGrp に記録される

5) 復元されたデータを、所定の場所にセットする（表示が編集前の状態に戻る）

```
if(GetLssFileType() == K_LSS_S) {//ジオメトリを編集中の場合
        scn = s3GetScene(GetSceneCurrentIndex());
        scn->mdl->root[0] = *enroParam.OrgGrp;
        g3SetRootGroup(scn->mdl->num, enroParam.OrgGrp);
} else if(GetLssFileType() == K_LSS_G) {//シーンを編集中の場合
        g3SetRootGroup(1, enroParam.OrgGrp);
}
```

④ 終了ボタンが押された場合

CParkRoadWnd::OnDestroy()の中で、

enroParam.SvGrp を解放する。

解放処理を個別に記述する代わりに u3RestoreGroupFamily 関数の一番目の引数を NULL、二番目の引数を SvGrp として呼び出すと、SvGrp に記述されたバックアップファイルを削除し、SvGrp 自体も破却する。

ここで、バックアップと復元に関連するデータをリスト９－１に示す。

リスト９－１：バックアップと復元に関連するデータと関数

```
enroParam のメンバ
d3Group**OrgGrp
void *SvGrp

typedef struct{
  char file[I3_MAX_PATHNAME];
  char gname[I3_MAX_PATHNAME];
}u3SaveGroup;

void*u3SaveGroupFamily(d3Group*g);
指定されたグループをファイルに保存する。
グループ名とファイル名を格納した u3SaveGroup 構造体（メモリブロック）を構築し、
そのアドレスを void 型へのポインタとして返す。

void*u3RestoreGroupFamily(int load, void*sginfo)
sginfo をクリアする（ファイルは削除し sginfo 構造体も解放する）。
load が非零なら、sginfo に記録されたファイルを読み込み、グループを返す。
      零なら、NULL を返す。
```

この方法は、プラグイン DLL であるトンネル生成機能(tunnel.dll)、法面生成機能

(nori.dll)、園路生成機能(ParkRoad.dll)、及び地形編集機能(land.dll)に含まれる、標高面作成、地形切断機能、頂点移動機能、地形の細分化と最適化機能で用いている。

９－３．移動・回転・スケール（CEditMove）

内部データとしては、選択したグループの上方リンクに付随するリンク・マトリクスを変更する操作である。編集操作に先立って、対象となるリンク・マトリクスを、BackupInitLayoutMatrix()関数(dataope.c)によりバックアップする。「戻す」ボタン、またはキャンセル終了により、バックアップされたマトリクスを、SetInitLayoutGroupLink()関数により復原する。

但し、編集が行われた後に、選択の変更（上位グループ、下位グループへ）が行われた場合には、新たに選択されたリンクのマトリクスが上書きバックアップされるため、復原することはできない。

９－４．マテリアル

マテリアルの編集は、モデルの編集の１種であるが、以下のような特殊な条件がある。
①ユーザーの便宜のために、編集ダイアログを開いたまま、複数のオブジェクトに対して次々と編集できること。
②オブジェクトの表面光学特性の編集のために用意されている複数のダイアログがあり、相互に呼び出しができること。

a. オリジナルのマテリアル編集画面(４－４(21)参照、CEditMate)
　編集画面ではカラーのみを直接編集する。マテリアル編集(b)、テクスチャ編集(c)のダイアログを下請け的に起動する。BackupGroup, GetBackupGroup, BackupFace, GetBackupFace の各関数を用いる。

b. マテリアル選択画面(４－４(24)参照、CMatList)
　マテリアル一覧をリスト表示し、ここから選択が行われた時点で、マテリアル編集画面(a)の上部の色球にのみ反映する。OK で終了した場合には、メイン画面で選択された対象物に反映する。キャンセルで終了した場合には、メイン画面には影響しない。上記のマテリアル選択操作でマテリアル編集画面に表示するマテリアルが、起動当初とは異なる表示となっていた場合であっても、こちらは復原しない。

c. オリジナルのテクスチャ編集画面(４－４(22)参照、CEditText)
　上記 a から起動するが、独自のバックアップ・復原を行う(BackupGroup, GetBackupGroup, BackupFace, GetBackupFace の各関数を使用)。このダイアログを OK で終了した場合、a の画面に戻るが、ここでキャンセルによる終了が行われてもテクスチャ編集結果を復原しない。

d. グラフィックなマテリアル編集画面（４－４(25)参照、CEditMaterial）
　マテリアルがリストボックスから選択された時点で、ダイアログ上部の色球に反映

する。上記 a からも起動可能であることから、特別なバックアップ処理を行わない。

e. グラフィックなテクスチャ編集画面（４－４(26)参照、CEditTexture）

テクスチャがリストボックスから選択された時点で、ダイアログ上部の図形に反映する。独自のバックアップ・復原処理を行う（BackupGroup, GetBackupGroup, BackupFace, GetBackupFace の各関数を使用)。

f. 様々な表色系によるカラー編集画面（４－４(28)参照、CColorSet）

編集開始に先立って、バックアップを作成する。その際、選択モードがグループであれば、BackupGroup() 関数(dataope.c)によりグループを、また面であれば BackupFace() 関数(dataope.c)により面をバックアップする。キャンセルで終了となった場合には、GetBackupGroup()関数、GetBackupFace()関数により復原を行う。

９－５．光源の編集

光源は、各シーンに対して一つずつ定義される光源グループ（LG、メモリ・ブロック）に、最大８の光源ユニット（メモリ・ブロック）が所属する構成となっている。光源グループは、複数のシーンから共有することができ、また光源ユニットも複数の光源グループ（従ってシーン）から共有できる、という複雑で重層的な構成となっている。

光源の編集においては、ＯＫボタンによる終了と、キャンセルによる終了があり、前者の場合にはそのままの状態で、また後者の場合には、編集前の状態に復原した上で処理を終了している。

編集処理を行うに先立ってバックアップを作成する。処理結果に対して、OK で終了すれば、バックアップを解放する。一方キャンセルで終了すれば、バックアップを復元すると共に、処理結果を解放する。解放に際しては、他のシーンから参照されている光源ユニット、光源グループに関するチェックを行わないと、リンク切れを起こす。例えば、ある光源ユニットの編集に際して、メモリブロックの再構成（realloc）が行われた場合、たとえ新たに取得された光源ユニットが同一名称をもち、編集対象としているシーンに固有の光源グループにリンクされていたとしても、他のシーンが使用している別の光源グループからの、この光源ユニットへのリンクは失われている。

① バックアップ

光源自動計算の場合には、既存の光源グループ LG のバックアップを BU 関数により作成する(gydlg.cpp)。この関数は、LSS-S（シーン）編集モードにおいては、編集対象である仮のシーン TempScene のメンバである光源グループのポインタをコピーする。また、LSS-G（モデル）の編集モードにおいては、唯一の光源グループのコピーを作成する。

② OK 処理

OK で終了する場合には、BB 関数によりバックアップされたデータを処理する。LSS-G モードにおいては、バックアップした光源グループ LG を解放する。LSS-S も

ードにおいては、LG への他のシーンからの参照をチェックし、他からも参照されていれば、そのままとする。唯一の LG であれば、他から参照されていない光源 L のみを解放した上で、LG を解放する。

③ キャンセル処理

キャンセルで終了する場合には、UB 関数により光源グループを復原する。それまでの編集で新たに作成した LG とそれに帰属する L を解放する。

LSS-S（シーン）の編集モードにおいては、光源を編集した結果は、シャッター操作により初めて確定し、シーン配列の中の特定の要素の中に TempScene からコピー・保存され、更に編集終了後のファイル保存に反映される。

LSS-G（モデル）の編集モードにおいては、光源編集は他の編集操作を容易化することだけが目的であり、処理結果をファイルに保存することはない。従って、処理結果は、g3SetLightGroup(&LG)関数で直接表示状態に反映する。この関数は、光源グループのアドレス(&LG)は保持せず、内部でコピーを作成して表示に使用する。このコピーの処理の中で、それまでの LG は解放される。従って、不適切な LG 情報が渡されると、その時点ではなく次の更新の中での解放処理の中で障害を生じる可能性がある点は注意を要する。更に、g3GetLightGroup()関数は、g3 空間にある現在の LG のポインタを返す。このポインタを用いて、不適切な処理（例えば、メモリーブロックの解放）が行われると、やはり以後の更新処理の中でエラーを生じる。従って、光源に関するポインタの処理は特に慎重に行わなければならない。

このような仕組になっている理由は、上述の複雑で重層的な LG 情報の管理方法にある。表示の LG 情報を変更したい時には、作成した LG 情報を g3SetLightGroup(&LG)関数で表示に反映させるが、作成した側で LG 情報を削除することができる。また、シーンの切換に際しては、シーン・リストに保持してあるオリジナル情報を s3SetLightGroup 関数で G3 空間にコピーするだけでよい。仮に G3 空間のコピーが変更（例えば削除）されても、オリジナル情報は影響を受けない。

９－６．効果

効果のデータ構造は、光源の場合に似て重層的である。即ち、一つのシーンに割り当てられた一つの効果グループが、効果ユニットを複数（効果の場合には総数に上限はない）参照できる。また、ひとつの効果ユニットは、複数の効果グループから共用される場合がある。また、光源グループの場合、ひとつのシーンが必ず一つの光源グループを持ち、しかも光源グループには最低一つの光源ユニットが無ければならない、という規則が存在するが、効果の場合にはこのような制約はなく、効果グループを特に持たないシーンや、存在意義は別として、効果ユニットがない効果グループなどが存在しうる。

従って、ひとつのシーンに対する効果の編集のバックアップと、キャンセルで終了した

場合のリカバリーは、全てのシーンに対して行っている。即ち、シーンの数と同数の効果グループの配列をバックアップ用に用意しておき、効果の編集開始時に各シーンの効果グループと効果ユニットのコピーをこれに割り当てる。効果ユニットは、同名で別のメモリブロックが発生すると、上記のデータ構造の一貫性が失われるため、リストとして管理し、既に同名の効果ユニットのコピーが作成されている場合には、新たに作成せず、これを参照する。

正常終了（OK）の場合には、このバックアップを全て解放し終了する。一方、キャンセル終了した場合には、現在の全てのシーンに登録されている効果グループと効果ユニットを全て解放したうえで、バックアップの効果グループと効果のポインタを各シーンにコピーする。

いずれの場合にも、効果グループと効果ユニットの解放に際しては、解放済みの効果ユニットを再度解放しようとして、アクセス違反を犯すことが無いように、「ごみ箱」に相当する効果ユニットの辞書（配列）をまず用意して、この上に解放すべき効果ユニットへのポインタの一覧を作成した上で、最後に各効果ユニットを一度だけ解放している。

９－７．処理方法の使い分け

バックアップ・アンドゥの処理方法には、以上に示したように、いくつかの異なるアルゴリズムがあり、現在のバージョンの中では、必要性や状況により使い分けている。これを大きく分けると、以下のように整理することができる。

①編集対象による違い

編集対象が地形や地物全体などの大きなモデルである場合には、バックアップをメモリ上に構築することはメモリ制約を受けるおそれがあるため、処理時間をある程度犠牲にして、処理前のモデルをファイルに保存しておき、もし処理結果がキャンセルされた場合には、処理結果を廃棄し、ファイルからモデルを再構築している。

編集対象が光源その他環境設定に関するものである場合、保存方法を統一化することは難しい。

②編集方法による違い

たとえば、簡単なダイアログで比較的単純な処理を行う場合には OK ボタンとキャンセル・ボタンを用意し、OK ボタンが押された場合のみ処理を行う。一度行われた処理の取り消しを行うことはできない。キャンセル・ボタンでは、処理を行うことなく終了する。

地形に係る編集などのように、実行ボタン、取り消しボタン、終了ボタンの３種類を有するダイアログでは、実行によりデータが変更される。キャンセル・ボタンで元のデータに復元する。終了ボタンで、処理実行の有無や結果の成否にかかわらず、その状態でダイアログを終了する。

カラー・マテリアルや光源の編集のような場合には、OK とキャンセルのボタン操作ではなく、ダイアログ上のリストからの選択やスライドバー操作により編集を行う。このため、

終了を OK ボタンで行うか、キャンセル・ボタンで行うかで、処理結果の採否を確定している。更に、作業能率のために、ダイアログを開いたまま、メイン画面で選択対象を変えて、それぞれのオブジェクトのカラーやマテリアル設定を「次々と」編集することを可能にしている。このような場合には、キャンセル処理で一つずつ復元することは複雑なデータ構造を必要とするため、キャンセル処理においては、ダイアログが開いた時点に復原している。