Les résultats en sortie: nouveau ou copie

Les résultats d'une étape peuvent être complètement nouveaux (ex. on crée une grille à partir de nuages de points) ou alors constituer une copie des intrants à laquelle on ajoute quelque chose (ex. ajouter une grille nb/nt aux grilles de nb, nt, densité, etc.). Ces deux options se répercutent dans les méthodes createInResultModelListProtected, createOutResultModelListProtected et compute.

Créer un nouveau résultat

Dans le cas où on crée un nouveau résultat de toutes pièces, on charge le modèle intrant en tant que CT_InResultModelGroup et on y ajoute les items nécessaires au traitement.

```
// Other includes...
#include "ct_result/model/inModel/ct_inresultmodelgroup.h"
#include "ct_result/model/outModel/ct_outresultmodelgroup.h"

// Alias for indexing models
#define DEF_in_res "in_result"
#define DEF_in_grp "in_group"
#define DEF_in_nb "nb (before)"
#define DEF_in_nt "nt (theoretical)"
#define DEF_out_res "out_result"
#define DEF_out_grp "out_group"
#define DEF_out_item "nbovernt"

// Other methods...

void LVOX_StepNbNtGrids::createInResultModelListProtected()
{
    CT_InResultModelGroup *in_res_model = createNewInResultModel(DEF_in_res, tr("Grilles"));
    resultModel->setZeroOrMoreRootGroup();
    resultModel->addGroupModel("", DEF_in_grp);
    resultModel->addItemModel(DEF_in_grp, DEF_in_nb, CT_Grid3D<int>::staticGetType(),
    tr("nb (before)"));
    resultModel->addItemModel(DEF_in_grp, DEF_in_nt, CT_Grid3D<int>::staticGetType(),
    tr("nt (theoretical)"));
}
```

On crée le modèle sortant en tant que CT_OutResultModelGroup et, puisqu'il s'agit d'un nouveau résultat, on doit y ajouter un groupe avant d'ajouter les items que produira l'étape.

```
void LVOX_StepNbNtGrids::createOutResultModelListProtected()
{
    CT_OutResultModelGroup *out_res = createNewOutResultModel(DEF_out_res, tr("nb/nt"));
    if (res != NULL)
    {
        res->setRootGroup(DEF_out_grp, new CT_StandardItemGroup(), tr("grille"));
        res->addItemModel(DEF_out_grp, DEF_out_item, new CT_Grid3D<float>(), tr("nb/nt"));
    }
}
```

Lors du traitement, on doit récupérer les résultats en entrée et en sortie. Pour créer les nouvelles structures de données, on utilise la chaîne de caractère représentée par DEF_out_item et le pointeur du résultat où les données seront placées.

```
void LVOX_StepNbNtGrids::compute()
{
    // Get the input
    CT_ResultGroup* in_res = getInputResults().first();

    // Get the out res and group
    CT_ResultGroup* out_res = getOutResultList().first();
    CT_StandardItemGroup* out_group = new CT_StandardItemGroup(DEF_out_grp, out_res);

    // Iterate
    CT_ResultGroupIterator it(in_res, this, DEF_in_grp);
    while (it.hasNext() && !isStopped())
    {
         // Get necessary input items
        CT_StandardItemGroup* in_group = (CT_StandardItemGroup*) it.next();
        const CT_Grid3D<int>* nb = (CT_Grid3D<int>*) in_group->firstItemByINModelName(this
, DEF_in_nb);
        const CT_Grid3D<int>* nt = (CT_Grid3D<int>*) in_group->firstItemByINModelName(this
, DEF_in_nt);

         // Create the new data structure
        CT_Grid3D<float>* nbnt = new CT_Grid3D<float>(DEF_out_item, out_res,
            nb->minX(), nb->minY(), nb->minZ(),
            nb->xdim(), nb->ydim(), nb->zdim(),
            nb->resolution(), nb->NA(), nb->NA());

         // Treatment...
         // ...

         // Add item to group to result
        out_group->addItemDrawable(nbnt);
        out_res->addGroup(out_group);
        nbnt->computeMinMax();
    }
}
```

Copier le résultat intrant

En revanche, si on veut ajouter au résultant intrant, quelques modifications doivent être faites au .h. Une variable privée de type CT_AutoRenameModels doit être ajoutée, ce qui requiert l'inclusion de l'entête correspondant. CT_AutoRenameModels est une classe propre à Computree qui garantit l'unicité du nom du modèle.

```
#include "ct_tools/model/ct_autorenamemodels.h"

// ...

class LVOX_StepNbNtGrids: public CT_AbstractStep
{
    Q_OBJECT
public:
    // ...

private:
    // ...

    CT_AutoRenameModels _nbnt_ModelName;
};
```

Le résultat en entrée étant destiné à une copie, on doit le charger en tant que CT_InResultModelGroupToCopy à l'aide de la méthode createNewInResultModelForCopy.

```
// Other includes...
#include "ct_result/model/inModel/ct_inresultmodelgrouptocopy.h"
#include "ct_result/model/outModel/tools/ct_outresultmodelgrouptocopypossibilities.h"

// Alias for indexing models
#define DEF_in_res "result"
#define DEF_in_grp "group"
#define DEF_in_nb "nb (before)"
#define DEF_in_nt "nt (theoretical)"

void LVOX_StepNbNtGrids::createInResultModelListProtected()
{
    CT_InResultModelGroupToCopy* in_res = createNewInResultModelForCopy(DEF_in_res, tr("Grilles"
));

    in_res->setZeroOrMoreRootGroup();
    in_res->addGroupModel("", DEF_in_grp);
    in_res->addItemModel(DEF_in_grp, DEF_in_nb, CT_Grid3D<int>::staticGetType(),    tr("
nb (before)"));
    in_res->addItemModel(DEF_in_grp, DEF_in_nt, CT_Grid3D<int>::staticGetType(),    tr("nt (theo)"
));
}
```

On crée le modèle sortant en tant que CT_OutResultModelGroupToCopyPossibilities à l'aide de la méthode createNewOutResultModelToCopy. On remarque que la chaîne de caractères à utiliser est celle du résultat entrant, que l'on copie. Puisque ce résultat est déjà bien formé, ajouter un groupe racine n'est pas nécessaire. L'item à ajouter est ensuite spécifié à l'aide de la variable de type CT_AutoRename.

```
void LVOX_StepNbNtGrids::createOutResultModelListProtected()
{
    CT_OutResultModelGroupToCopyPossibilities* out_res = createNewOutResultModelToCopy(DEF_in_res)
;
    if (out_res != NULL)
    {
        out_res->addItemModel(DEF_in_grp, _nbnt_ModelName, new CT_Grid3D<float>(),tr("ndovernt"));
    }
}
```

Au cours du traitement de données, on récupère d'abord les résultats en sortie, qui contiennent une copie des résultats en entrée. Pour créer les nouvelles données, une chaîne de caractères est nécessaire (représentée par le #define dans l'exemple "Créer un nouveau résultat"); on utilise donc la méthode CT_AutoRenameModels::completeName.

```
void LVOX_StepNbNtGrids::compute()
{
    // Get the output res, which is a copy of input result
    CT_ResultGroup* out = getOutResultList().first();

    // Iterate
    CT_ResultGroupIterator it(out, this, DEF_in_grp);
    while (it.hasNext() && !isStopped())
    {
         // Get the input group copy
        CT_StandardItemGroup* group = (CT_StandardItemGroup*) it.next();
         // Get inputs for treatment...
         // ...
```

```
            // Create the new data structure
            CT_Grid3D<float>* nbnt = new CT_Grid3D<float>(_nbnt_ModelName.completeName(), out,
                nb->minX(), nb->minY(), nb->minZ(),
                nb->xdim(), nb->ydim(), nb->zdim(),
                nb->resolution(), nb->NA(), nb->NA());

            // Treatment...
            // ...

            // Add the new data structure to the input group copy
            group->addItemDrawable(nbnt);
        }
}
```