

Système de formules

Vue d'ensemble

Le système de formules permet à un étudiant de calculer automatiquement la valeur d'une cellule à partir d'autres cellules numériques d'un tableau de module. Une formule est une liste de références signées : chaque référence est ajoutée ou soustraite du total.

Le calcul est effectué dans Unity, tandis que la définition de la formule est persistée côté backend. La valeur calculée reste stockée dans `DataDrivenInputData.value` ; la structure de la formule est stockée séparément dans `DataDrivenInputData.formula`.

1. Architecture générale

```
flowchart LR
    USER[Étudiant] --> EDITOR[DataDrivenTableEditorControl]
    EDITOR --> TABLE[DataDrivenTableControl]
    TABLE --> CELL[InputCellControl]
    TABLE --> FORMULA[DataDrivenInputFormulaData]
    FORMULA --> REF[DataDrivenInputFormulaReferenceData]
    TABLE --> CALC[Calcul + graphe de dépendances]
    CALC --> VALUE[DataDrivenInputData.value]
    VALUE --> CELL
    TABLE --> CTRL[TeamByModuleController]
    CTRL --> CMD[save_student_module_input_formula]
    CMD --> BACK[Backend Node.js / BrainCloud]
    BACK --> STORE[(teamsModules)]
    BACK --> EVENT[team_module_input_formula_changed]
    EVENT --> CTRL
    CTRL --> TABLE
```

2. Modèle de données

Une cellule peut contenir une formule optionnelle. La formule contient une liste de références; chaque référence identifie complètement une cellule et contient son signe.

```
classDiagram
class DataDrivenInputData {
    string value
    Dictionary options
    DataDrivenInputFormulaData formula
}
class DataDrivenInputFormulaData {
    List references
    Serialize()
    Deserialize()
    Clone()
}
class DataDrivenInputFormulaReferenceData {
    string pageUid
    string tableUid
    string sectionUid
    string rowUid
    string inputUid
    int sign
}
DataDrivenInputData --> DataDrivenInputFormulaData
DataDrivenInputFormulaData --> "0..*" DataDrivenInputFormulaReferenceData
```

Adresse complète d'une référence

Un `inputUid` seul n'est pas suffisant, puisque le même input peut exister dans plusieurs sections ou rangées. Une référence conserve donc `pageUid`, `tableUid`, `sectionUid`, `rowUid`, `inputUid` et `sign`. Le signe vaut `1` pour additionner et `-1` pour soustraire.

3. Création d'une formule dans l'interface

```
sequenceDiagram
actor U as Étudiant
participant E as DataDrivenTableEditorControl
participant T as DataDrivenTableControl
participant C as InputCellControl
U->>C: Sélectionne la cellule résultat
U->>E: Active le mode formule
E->>T: onFormulaModeValueChanged(true)
T->>T: signe initial = PLUS
T->>C: met les cellules compatibles en mode sélection
U->>E: choisit + ou -
E->>T: onFormulaReferenceSignChanged(sign)
U->>C: clique une cellule référence
C->>T: onSelected(inputUid)
T->>T: SelectFormulaReference(...)
T->>T: ajoute / change / retire la référence
T->>C: outline vert (+) ou rouge (-)
U->>E: désactive le mode formule / Enter
E->>T: onFormulaModeValueChanged(false)
T->>T: sauvegarde la formule
```

Comportement lors d'un clic sur une référence

- si la référence n'existe pas, elle est ajoutée;
- si elle existe avec l'autre signe, son signe est remplacé;
- si elle existe déjà avec le même signe, elle est retirée;
- la cellule résultat ne peut pas se référencer elle-même;
- seules les cellules numériques compatibles sont sélectionnables en mode formule;
- une cellule qui dépend directement de la cellule résultat est rendue non sélectionnable afin d'éviter une dépendance circulaire directe.

Les références positives sont affichées avec un contour vert et les références négatives avec un contour rouge via `InputCellControl.SetFormulaOutline()`.

4. Calcul d'une formule

```

flowchart TD
  F[DataDrivenInputFormulaData] --> LOOP[Pour chaque référence]
  LOOP --> GET[TryGetInputData]
  GET --> PARSE[Parse inputData.value en double]
  PARSE -->|échec| ZERO[Valeur = 0]
  PARSE --> MOD{value_modifier présent ?}
  ZERO --> MOD
  MOD -->|Oui| APPLY[referenceValue *= valueModifier]
  MOD -->|Non| SIGN
  APPLY --> SIGN[Appliquer sign +1 / -1]
  SIGN --> SUM[Ajouter au total]
  SUM --> LOOP
  LOOP -->|fin| VALUE[Écrire le résultat dans inputData.value]
  VALUE --> UI[Mettre à jour InputCellControl]
  VALUE --> TOTAL[Mettre à jour total row / journal si applicable]

```

Les valeurs impossibles à parser sont traitées comme `0`. Si l'option serveur `value_modifier` est présente sur une cellule référencée, elle est appliquée avant le signe de la référence.

5. Propagation des dépendances

Le système construit un index inverse des dépendances dans `_formulaListenersByReference`. La clé d'une cellule est construite sous la forme `sectionUid|rowUid|inputUid`.

```

flowchart LR
  A[Cellule A change] --> LOOKUP[_formulaListenersByReference A]
  LOOKUP --> B[Formule B dépend de A]
  LOOKUP --> C[Formule C dépend de A]
  B --> RB[RecalculateFormula B]
  C --> RC[RecalculateFormula C]
  RB -->|valeur changée| NEXTB[Notifier les formules dépendantes de B]
  RC -->|valeur changée| NEXTC[Notifier les formules dépendantes de C]
  NEXTB --> D[Formule D]
  NEXTC --> E[Formule E]

```

La propagation est récursive : lorsqu'une formule change de valeur, les formules qui dépendent d'elle sont recalculées à leur tour. Un `HashSet` de clés déjà visitées empêche la propagation de boucler indéfiniment pendant un même cycle de notification.

Initialisation

Lors du chargement ou d'un refresh important, `InitializeFormulasByReference()` reconstruit entièrement l'index inverse, puis `RefreshFormulasByReferenceAndRecalculateAll()` recalcule toutes les cellules possédant une formule.

6. Sauvegarde et synchronisation backend

```
sequenceDiagram
    participant T as DataDrivenTableControl
    participant C as TeamByModuleController
    participant B as Backend Node.js
    participant DB as teamsModules
    participant M as Messages équipe/prof
    T->>C: SaveTeamModuleInputFormula(..., formula)
    C->>B: custom command save_student_module_input_formula
    B->>B: localise page/table/section/row/input
    alt formule avec références
        B->>DB: inputData.formula = formula
    else formule vide
        B->>DB: delete inputData.formula
    end
    B->>B: updateFormulaReferences(...)
    B->>DB: saveDocumentData
    B->>M: team_module_input_formula_changed
    M-->>C: message temps réel
    C-->>T: OnTeamModuleInputFormulaChanged(...)
    T->>T: reconstruit listeners + recalcule
```

Le backend persiste uniquement la définition de la formule. Si la liste de références est vide, la propriété `formula` est supprimée de l'input.

7. Sections et rangées dynamiques

Une section ou une rangée temporaire peut recevoir un nouvel UID lors de sa création côté serveur. Dans ce cas, les références de formules existantes doivent suivre le nouvel identifiant.

`utilsByModule.updateFormulaReferences()` parcourt les inputs de la table, des sections et des rangées et remplace les anciens `sectionUid`/`rowUid` par les UID persistés.

Lorsqu'une rangée ou une section est supprimée, les références qui pointent vers celle-ci sont retirées. Si une formule ne possède ensuite plus aucune référence, la formule est supprimée.

8. Activation et tutoriel

Le système est activé par la préférence `has_formulas`. Lorsque cette option est désactivée, les contrôles de formule ne sont pas affichés.

Deux séquences de tutoriel existent : `ModuleFormulaTutorial` introduit l'activation du mode formule, et `ModuleFormulaReferencesTutorial` explique les boutons + et - et la sélection des références.

Fichiers principaux

- `Assets/Src/UI/SimByModule/DataDrivenTableControl.cs` — logique principale, calcul et dépendances.
- `Assets/Src/UI/SimByModule/DataDrivenTableEditorControl.cs` — toggle formule et boutons + / -.
- `Assets/Src/UI/SimByModule/UI/InputCellControl.cs` — interaction cellule et contours visuels.
- `Assets/Src/Game/Data/ByModule/DataDrivenInputData.cs` — valeur calculée et formule optionnelle.
- `Assets/Src/Game/Data/ByModule/DataDrivenInputFormulaData.cs` — liste des références.
- `Assets/Src/Game/Data/ByModule/DataDrivenInputFormulaReferenceData.cs` — adresse complète et signe.
- `Assets/Src/UI/SimByModule/Controllers/TeamByModuleController.cs` — sauvegarde et réception de l'événement.
- `Assets/Src/UI/SimByModule/Tutorial/ModuleFormulaTutorial.cs` — tutoriels.
- `master/node.js/.../utils_custom_command_by_module.js` — persistance backend.
- `master/node.js/.../byModule_utils.js` — mise à jour/suppression des références lors des changements d'UID.

Points d'attention

- Le résultat numérique est dans `value`; la formule ne remplace pas ce champ.
- Une formule doit utiliser l'adresse complète de la cellule, pas seulement `inputUid`.

- Toute modification d'UID dynamique doit mettre à jour les références persistées.
 - Lorsqu'une cellule référencée change, il faut notifier les dépendances pour maintenir les formules en cascade.
 - Lorsqu'une formule est modifiée depuis un autre client, l'événement temps réel doit reconstruire ses listeners avant le recalcul.
-

Revision #1

Created 2026-09-17 21:13:35 UTC by Mathis

Updated 2026-09-17 21:18:25 UTC by Mathis