Dev Notes

Login avec le token de cheneliere

Pour se logger avec le token de cheneliere, le lien va d'abbord pointer vers clogin avec le user, le token, l'expiration et le isbn:

cheneliere.uniksim.com/clogin?username=test@iplusinteractif.com&token=59306298493CBB05D3F

Dans LoginController::cheneliereAutologin

- Le server (uniksim/master) poke le serveur cheneliere pour savoir les produits de ce user.

- Le serveur node.js est poker sur le port 8085 pour créer le user GameSparks

- si c'est bon, on l'envoie vers le bon jeu avec le ISBN, et on lui donne le token et l'expiration

Dans le jeu, on va poker notre serveur avec

getcodeInfo/{productId?}/{token}/{expiration}
Notre server repoke cheneliere pour etre sur que ce user est bon, et envoie les login info au jeu,
dans LoginController::getLoginInfo

Implémenter un nouveau jeu

Pour faire un nouveau jeu dans la structure il faut:

- Dupliquer la structure de dossier dans Assets\GameSpecific\Ent1
- Intégrer le jeu dans le login control
- Mettre [gameUid]_login_subtitle et [gameUid]_login_title dans sharedText
- Faire le skin pour le jeu
- Faire les Doozy game spec
- Faire les scenes de loading pour le jeu
- Faire le static data du jeu
- Faire les commandes spécifiques sur le serveur
- ajouter le gameUid dans server.js
- Faire la génération de résultats spécifique
- Faire l'évaluation spécifique.
  - Scenario
  - Segment (if needed)
  - SimPerf
    - Ajouter les pondérations dans les Google Sheets

Scenario (Client)

- Add **ScenarioPages** in GameSpecific/{GameUid}/Resources/data_{GameUid}
  - Create **ScenarioSegGridControlBase** if needed
- Fill the Google Sheets
  - Do not forget to export the data

Informations sur les jeux

Mk1

- Score du produit
  - Éthique ou non (3 pts)
  - Couleur (1 pts)
  - Type de matériaux (1 pts)
- Score de communication
  - Média 1 (1 pts)
  - Média 2 (1 pts)
- Score du budget de communication
  - Budget / budget moyen dans le segment
- Score de distribution
  - Type (1 pts)
  - Intensité (1 pts)
- Score du budget de distribution
  - Budget / budget moyen dans le segment
- Bonus de l'année de création du produit
  - 0 année depuis la création -> Bonus de 0.75
  - 1 année depuis la création -> Bonus de 1.1
  - 2 année depuis la création -> Bonus de 0.5
- Score du prix
  - Prix moyen du segment / prix de vente
-

Score d'un produit lancé

- ((Score du produit + Score de communication + Score du budget de communication + Score du budget de distribution + Bonus de l'année de création du produit) / 5) * score du prix * score de distribution

- Calcul du nombre de produit vendu
  - Calcul le score de chaque produit lancé dans un même segment
  - Calcul le nombre maximal de vente possible
    -

Mk2

Ent1

Hr1

Appartenance -> sur chaque type d'employé

- Score de chaque année et s'ils sont là depuis longtemps
  - Moyenne du taux de rétention depuis le début (cap 80%)
  - taux appartenance pour année -> renvoyer / employée totaux
  - cas (10 employé, 2 employés partent, 2 congédié (taux d'appartenance 60% qui est monté à 80% puisque minimum))

Mgt1

Fix le changement d'email de chenelière

PhpMyAdmin

1. Renommmer le "email" dans la table "users"
2. Modifier le "code" dans "products_code"
3. Dans gamesparks, changer le user aussi relié au compte
    1. mettre l'acien user dans query: { "userName" : "shirleydgrace@yahoo.com_chen_4191" }
    2. et le changer pour le nouveau
4.

Modifier le mot de passe du user

1. Se connecter en tant que la personne sur test harness
2. { "@class": ".ChangeUserDetailsRequest", "newPassword": "{new-email}_chen_{products_code:oid}" }

https://docs.gamesparks.com/tutorials/cloud-code-and-the-test-harness/changing-player-passwords.html

ByYear vs ByModule

Generation Mode

- ### ByYear
  - Bouton : Générer pour toutes les équipes de la simulation.
  - Calendrier : Générer pour toutes les équipes lorsque le champs "nextGenerate" est passé.
- ### ByModule
  - Bouton : Génère le module selectionné pour toutes équipes qui n'ont pas de temps supplémentaire et set le nextGenerate (situé dans "teamsModules") au temps désiré pour les équipes avec du temps supplémentaires.
  - Calendrier : Génère le module selectionné lorsque le "nextGenerate" est atteint. Les équipes avec du temps supplémentaires sont généré plutard (nextGenerate + temps supplémentaire).

Results generations

ByYear

- est calculé à partir du yearData (situé dans "teams").

ByModule

- est calculé à partir de "teamsModules".
  - Le seed permet d'avoir des résultats différents pour chaque équipe

Saved results

- Dans les deux types, les résultats sont sauvés partiellement dans "teams" (seulement les résultats d'une équipe) et dans "results" (les résultats de toutes les équipes).

VS

- Le code est séparé par le namespace "SimByModule" et "SimByYear" afin qu'une simulation n'est pas accès aux informations spécifique de l'autre type.

- Classe (SimulationByYear)
  - Contient le status de l'année courante (started, generating, finished).
  - Des évènements par année.
  - Présence par année pour un étudiant.
  - Année de pratique et réouvrir une année

- Classe (SimulationByModule)
  - Permets d'ajouter module à la simulation ainsi que de le commencer et finir.
  - Ajouter du temps supplémentaires à une équipe sur un module.

Unity UI

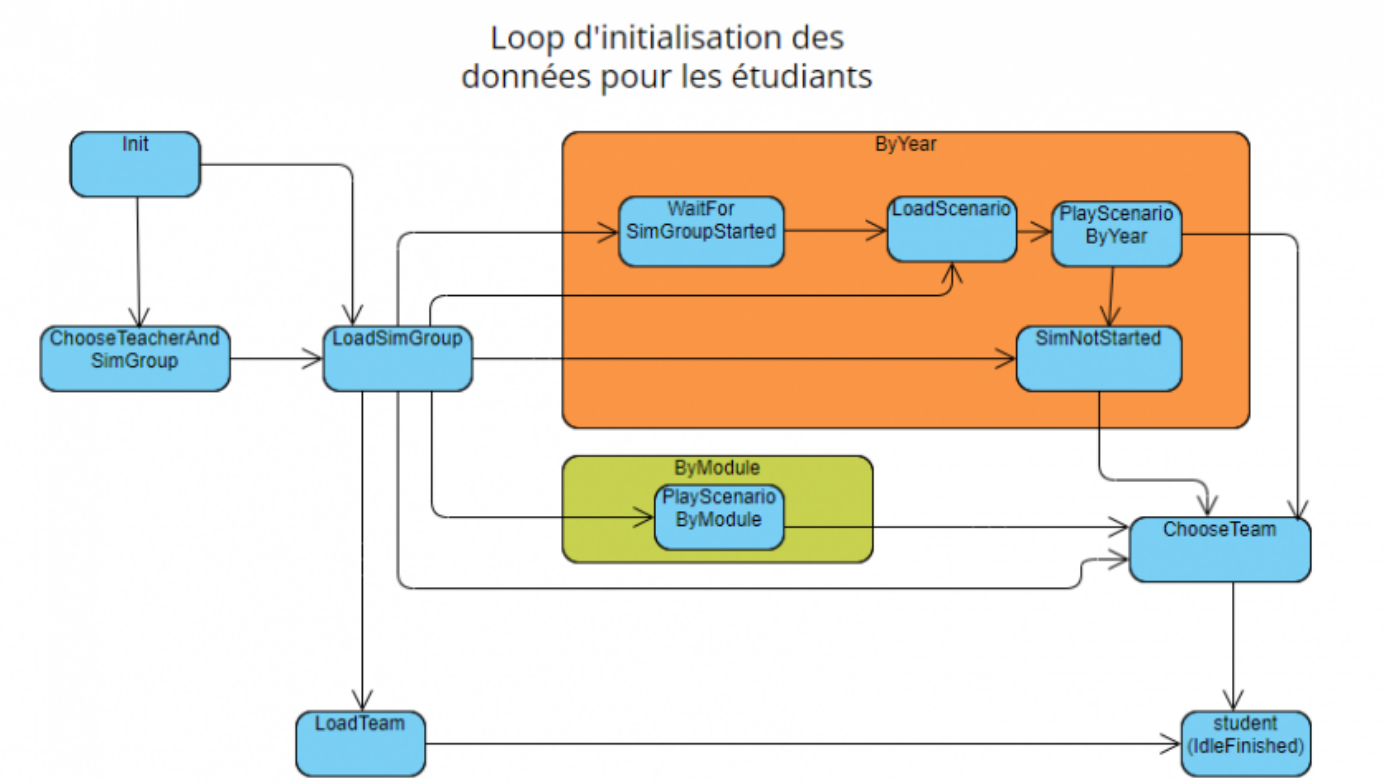
Création de simulation

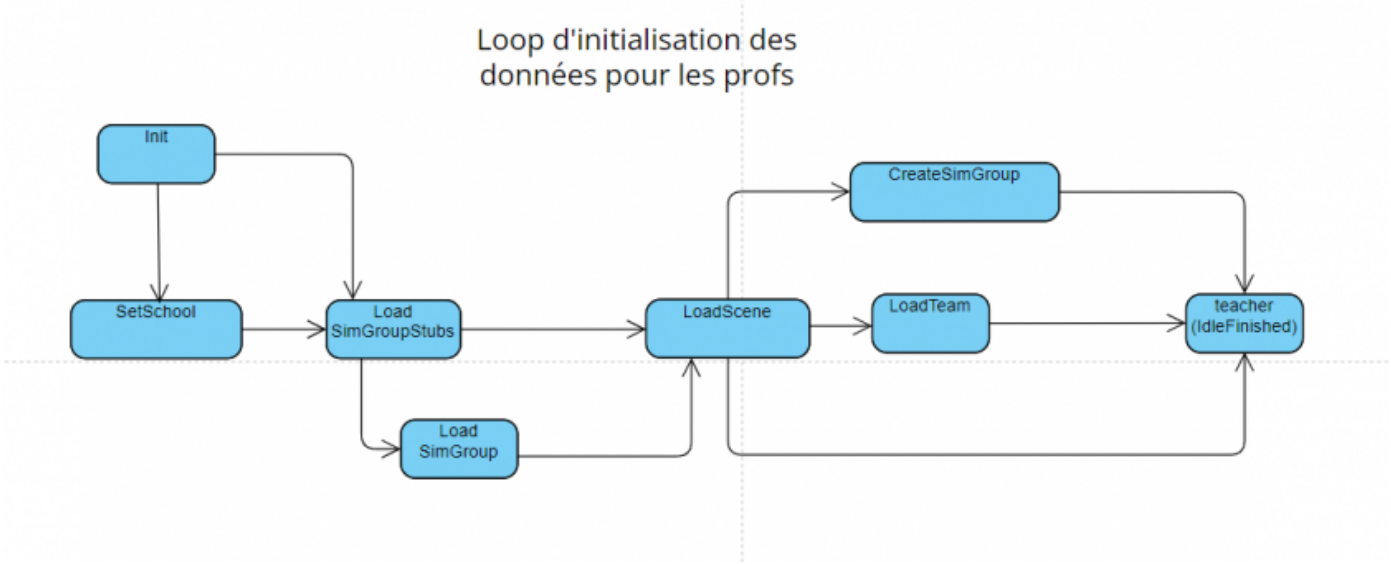
- ByYear à un champ supplémentaire (Nombre maximum de générations).

Vu Global

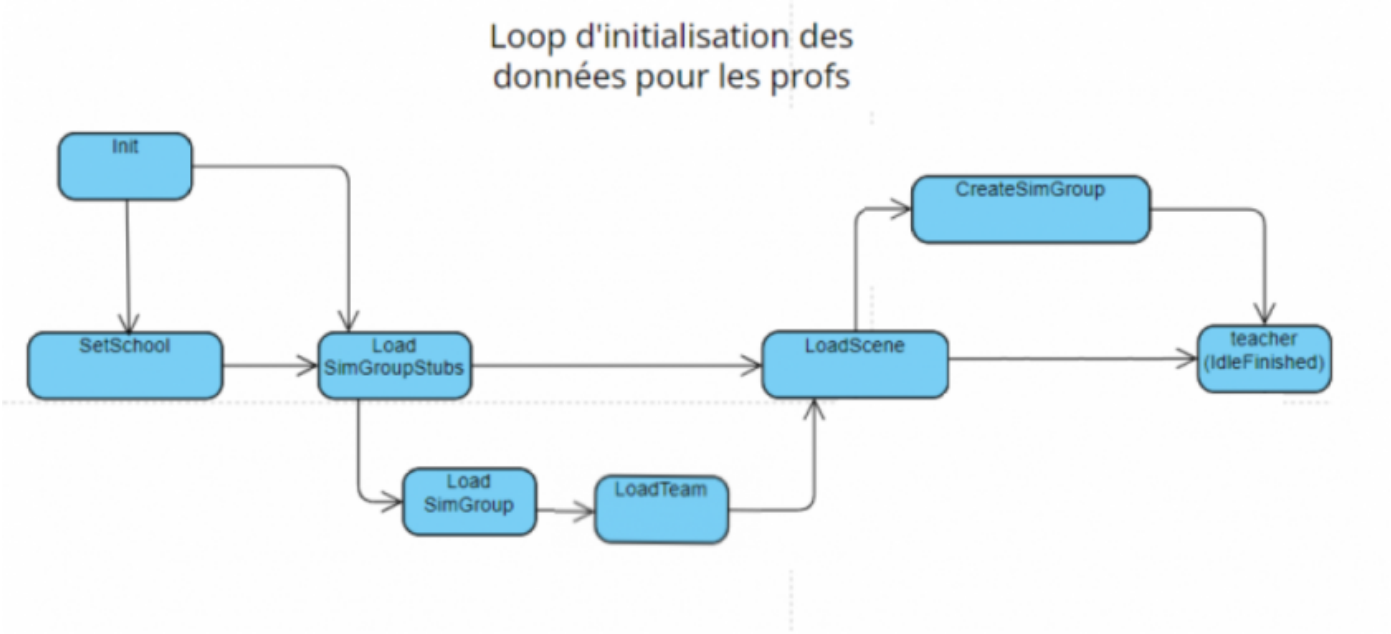
- ### ByYear
  - Contient le nombre d'année ainsi que l'état de chaque année.
- ### ByModule
  - Contients une liste des modules actifs ainsi que l'état de chacun.

Loop d'initialisation des données dans Unity




Loop d'initialisation des
données pour les profs
Init
SetSchool
Load
SimGroupStubs
Load
SimGroup
LoadScene
CreateSimGroup
LoadTeam
teacher
(IdleFinished)

Version 2.0


Loop d'initialisation des
données pour les profs
Init
SetSchool
Load
SimGroupStubs
Load
SimGroup
LoadTeam
LoadScene
CreateSimGroup
teacher
(IdleFinished)

Unit Testing ACC1

Pour unit tester ACC1, il faut décomposer chaque module et chaque fonction en bout de code testable avec un input et un output attendu.

Chaque module a 4 étapes distinctes:

1. Generate module type data
2. Answer Key
3. Grading
4. Re-Assemble (un grading post-process)

Par exemple, on veut tester le module de remises de taxes, il faut commencer par voir ce qui se passe dans ce module.

On commence par généré une commande dans unity qui nous redémarre le module en pratique.

```
data = {
        "commandUid":"restart_student_module",
        "commandData": {
            "teamUid":"be86465e-a8bf-42e5-bef1-447e94acc461",
            "actUid":"91dc1bce-d867-4c27-872d-84bd41d56ae9",
            "actModuleUid":"45d718be-2cab-4dcf-9328-737b46ed5ccd",
            "moduleType":"practice",
            "lastPraticeGrade":"-1"
        }
    };
```