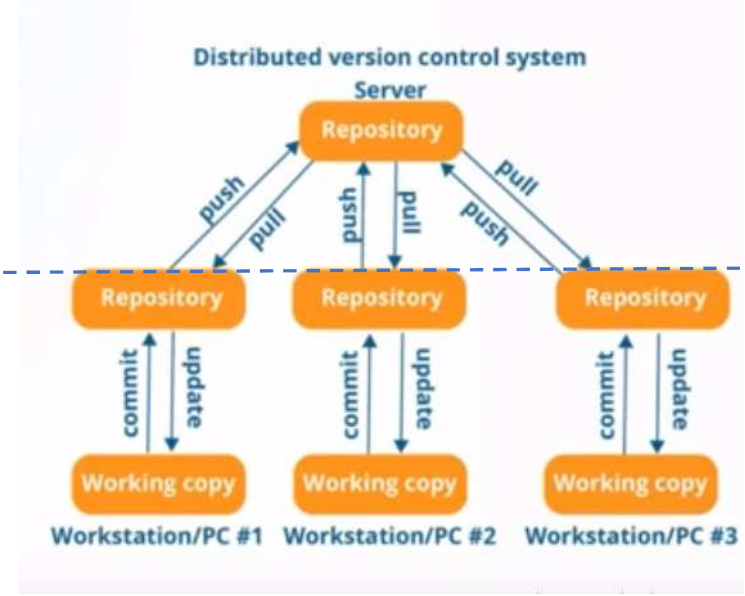
Travail collaboratif avec github

Le principe :

Github (tel que gitlab, bitbucket….) est un service pour héberger des dépôts = répertoires pour projet = Repository.

Git est un outil qui permet sur votre machine de créer un dépôt local lié et similaire au répertoire hébergé : repository. Vous pouvez alors envoyer vos mises à jour faites en local sur le repository grâce à des actions de **push** et inversement, vous pouvez récupérer depuis le serveur des mises à jour faites par les autres développeurs sur votre dépôt local grâce à des actions de **pull.** Il permet donc le travail collaboratif mais aussi le « versionning ».



Sur le serveur

En local

Toutes les actions push, pull, commit peuvent être déclenchées en ligne de commande :

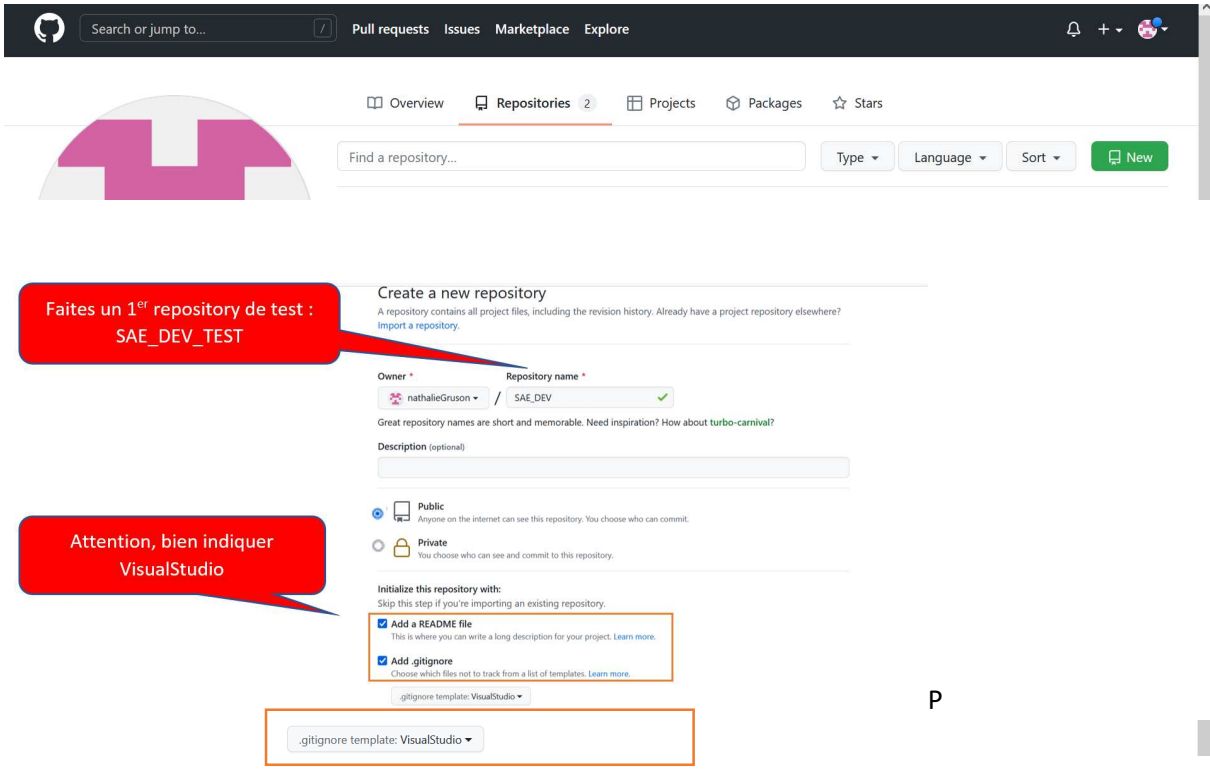
```
$ git pull
```

Mais Visual studio a une interface graphique intégrée pour déclencher les commandes git, ce que nous utiliserons dans ce tuto.

Etape 1 : Création d'un compte : chaque membre du groupe crée un compte sur Github avec son mail universitaire.

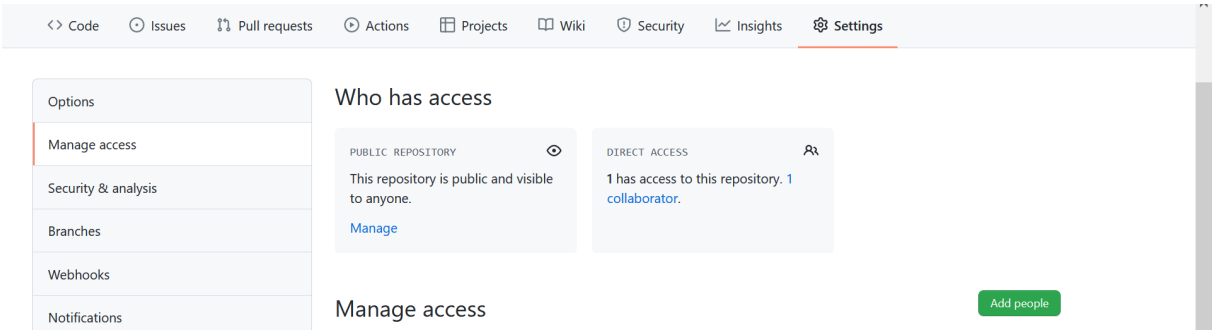
<u>Etape 2 : Création du repository : A FAIRE PAR UN SEUL ETUDIANT (les autres observent et vérifient !</u>

1. Un membre du groupe crée un répertoire de travail (repository)sur Github.



P

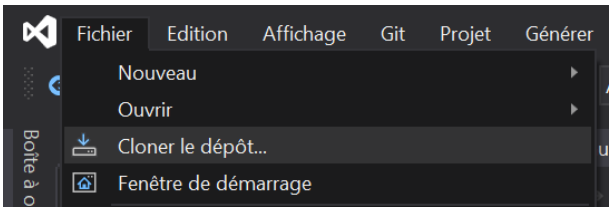
2. Invite les autres membres du groupe : Menu : Settings/Manage Access



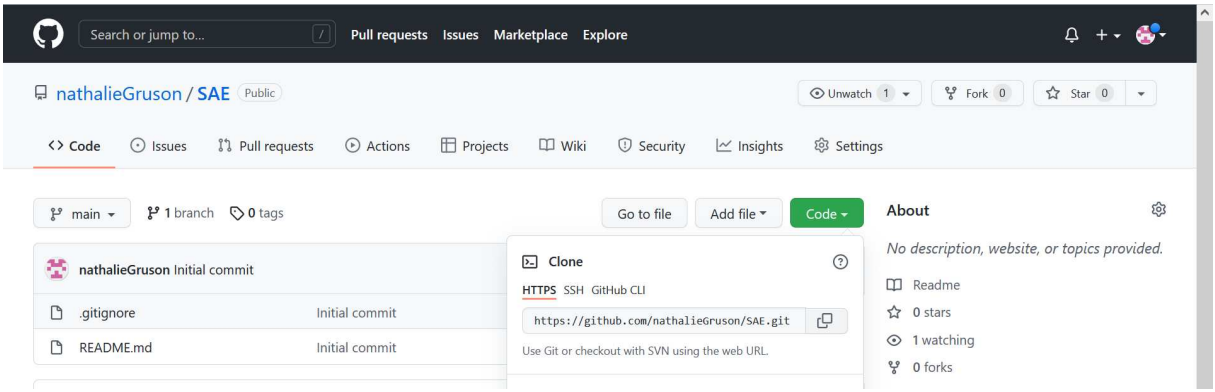
3. Depuis Visual Studio, le même étudiant clone le repository (Cela crée un répertoire local sur la machine de l'étudiant reliée au repository) :

Remarque importante utilisation à l'IUT :
Soit vous clonez sur le répertoire par défaut sur le C:\Users\ngrus\Source\Repos
Et le chemin relatif vers les dll sera toujours correct car elles sont dans le même répertoire : C:\Users\ngrus\.nuget\packages
Il faut bien évidemment être sûr d'avoir les dll sur le c :
Il faudra alors à chaque début de séance Cloner pour récupérer la version serveur et surtout tout envoyer au serveur à la fin de la séance pour ne rien perdre.

Soit vous clonez sur votre P : et il faudra alors recopier les packages sur votre P :

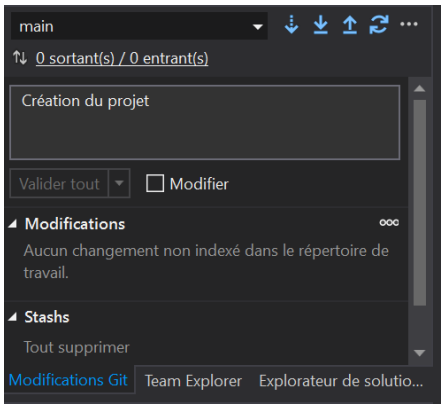


Il faut connaître l'url de stockage du repository : à récupérer dans le menu Code sur github



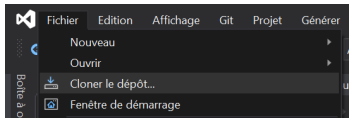
4. Il crée le projet monogame à l'intérieur de ce nouveau répertoire.

5. Il valide ses modifications pour les envoyer sur github : Pour cela, Dans l'onglet « modifications git », il renseigne le message. Ex : « création du projet » puis clique sur « valider tout » puis sur le flèche bleue montante à l'info bulle :« envoyer »



Etape 3 : Création des copies locales du repository : A FAIRE PAR LES AUTRES MEMBRES DU GROUPE

Depuis Visual Studio, les étudiants étudiant clone le repository (Cela crée un répertoire local sur la machine de l'étudiant relié au repository) :

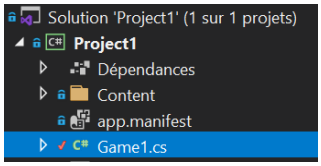


Il faut connaître l'url de stockage du repository : à récupérer dans le menu Code sur github

Etape 4 : Travailler ensemble « sans conflit »

Travailler ensemble sans conflit : c'est travailler sur 2 portions de code bien distinctes ! et donc avec un minimum de répartition et de communication.

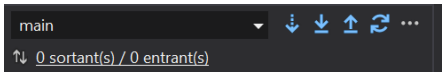
Ouvrez le .Sln, codez, modifiez . Pour tester : un étudiant fera une modif dans Initialize (Ex : Window.Title ="Test" ;) et un autre fera une modif dans Draw (Ex : `GraphicsDevice.Clear(Color.Red);`)



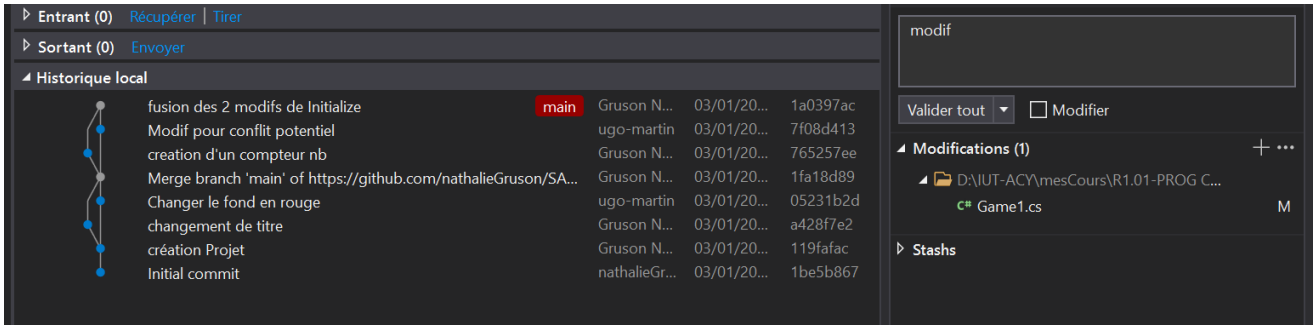
Au niveau de l'explorateur de solution, une coche rouge apparaît sur le fichier Game1 (indiquant une modification en attente), il faut alors :

Envoyer vos mises à jour : pour cela, il faut faire 3 actions dans l'onglet « Modifications git »

1. Valider ses modifs (« Commit »): Ecrire un message (Ex : Modification du titre de la fenêtre, Modification de la couleur de fond ») puis « Valider tout »
2. Toujours Récupérer (=Tirer) la version sur le serveur : 2eme flèche descendante (si conflits, les résoudre, valider la fusion avec un message)
3. Puis Envoyer , flèche montante .



Observez si vous avez bien récupéré les modifs de l'autre. Vous pouvez aussi cliquer sur la ligne « sortants/entrants » elle indique le nb de push (envois = sortants) et le nb de pull (tirage = entrant) en attente. Cela vous donnera l'historique de toutes les modifications validées ainsi que les auteurs des modifications



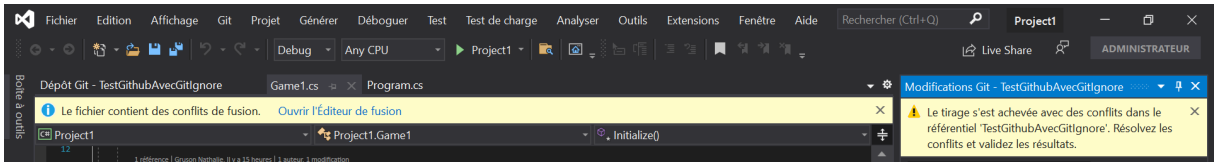
Récupérer les mises à jour des autres :

Cliquez sur Tirer : 2eme flèche descendante

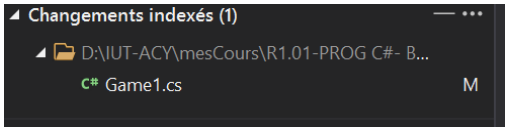
Etape 5 : Résoudre un conflit !

Vous êtes 2 ou plus à avoir modifié la même zone. Remarque : c'est une situation à éviter grâce à une répartition du travail réfléchie et la communication, mais cela pouvant arriver malgré tout, il faut savoir y remédier. Le 1er qui envoie n'aura pas de problème. Par contre, le ou les autres auront un conflit :

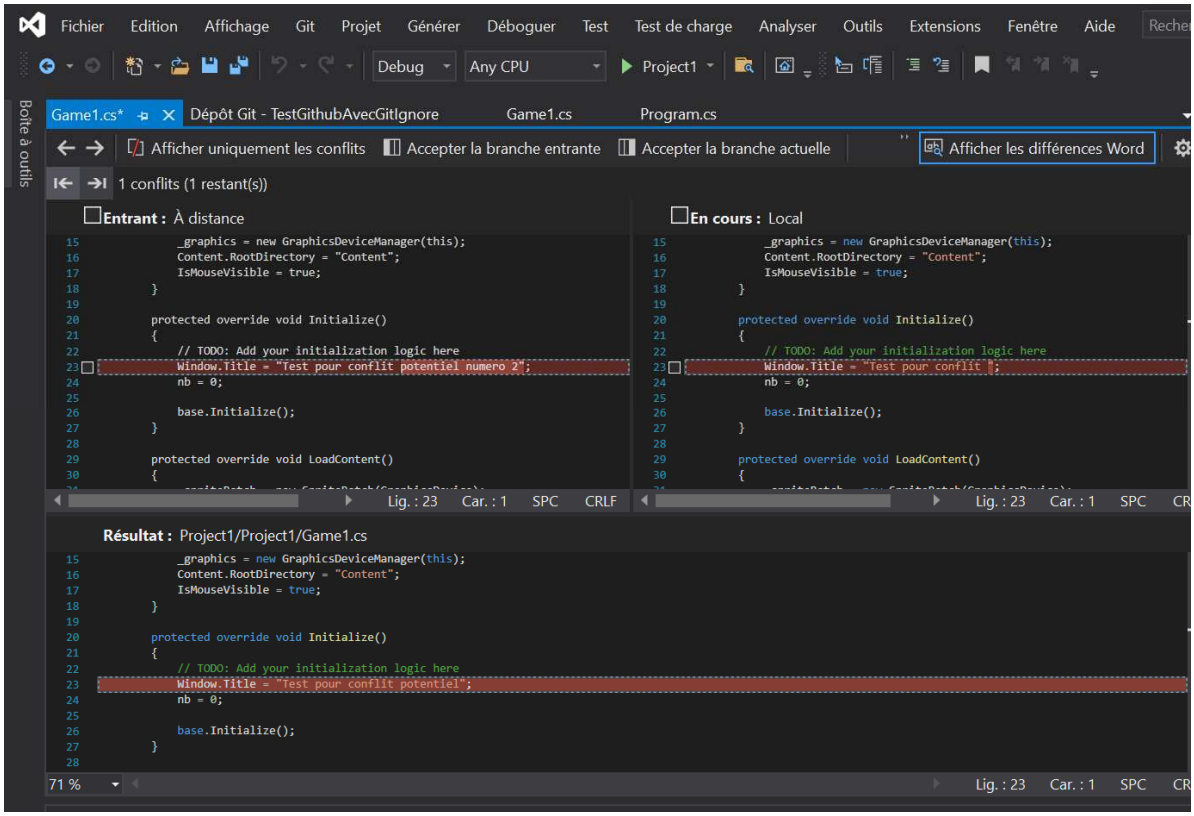
il faut alors :



1. Résoudre le conflit : Double-Cliquez sur le fichier en conflit : cela ouvre un éditeur de fusion : il vous montre :



   - à gauche ce qui vient du serveur (et donc la modif du 1er étudiant qui a envoyé ses mises à jours)
   - à droite votre version en locale
   - en bas, le résultat de la fusion



   Vous pouvez décider en cochant ou en cliquant en haut sur « Accepter la branche entrante ou actuelle », de prendre le code de gauche, de droite ou les 2, ou modifier directement dans la fenêtre résultat.

2. Accepter la fusion
3. Valider (avec un message)
4. Puis Envoyer