Emmanuel Ponti et Justin Pessia

Groupe 139

Journal

Notre Progression

Tache	Durée en heure~
Vecteur finis (pleinement opérationnels et testés)	4
Particules simples (pleinement opérationnels et testés)	3
Enceintes	1
Dessinable + SupportADessin	4
Systeme(pleinement opérationnels et testés)	1
exerciceP9 (simulation textuelle)	4
Graphisme : cadre général (reprise des exemples tuto)	4
Graphisme : adaptation à ce projet spécifique	4
Amélioration du modèle	5
Collisions avancées	3
fichier CONCEPTION	2
fichier README	1

Le suivi

Semaine 2 :
Premiers essai de poo en implémentant les vecteurs 3D pour l'instant dans un simple fichier .cc
Nous n'avons pas encore lu la mini référence sur les make file et cela n'a pas l'air si évident
Nous avons cependant commencé a lire le tutoriel graphisme qui prend pas mal de temps

Semaine 3 :
Nous avons commencé à séparer les fichiers en .h et .cc et copié collé le makefile en adaptant avec nos noms de fichier
Pour l'instant les taches hebdomadaires sont assez rapides donc nous le faisons ensemble

Semaine 4 :
Emmanuel: J'ai tout fais jusqu'à la question P4.3 : constructeurs et comparaison et opérateur de comparaison, j'ai aussi décidé d'implémenter le constructeur de copie voir fichier réponse

Justin: J'ai complété ce qui restait à faire mais je ne vois pas vraiment a quoi va servir le vecteur unitaire, je l'ai quand même mis on verra bien..

Semaine 5 :
Apres s'être réparti les taches (on a fait simple : une classe chacun) voici les observations que nous avons tiré de cette semaine:

Justin:

Il a fallu créer la classe Particule avec beaucoup d'attention, car celle-ci n'est en fait pas une sous classe de Vecteur3D car une particule n'est pas un vecteur, en revanche ses instances le sont, ce à quoi je n'avais pas pensé immédiatement. La difficulté pour moi s'est ensuite surtout présenté sur la fonction affiche() car j'avais totalement oublié l'existence de l'autre fonction affiche de la classe Vecteur3D qui permettait vraiment de faciliter le travail. Pas de problème avec le makefile que j'ai modifié sans difficulté, en revanche le retour du main de testParticule.cc ne me convient pas: une

masse avec 6 décimales est assignée a la particule p1 mais après compilation je récupère une précision de 4. J'ai alors essayé de palier à ce problème en utilisant la bibliothèque <iomanip> et setprecision mais cela ne me convient toujours pas car cela rajoute des 0 inutiles aux autres valeurs. A voir si je trouve la solution, sinon j'attendrais la prochaine seance d'assistanat, aussi je me dis que cela vient peut être du compilateur car certain ont une précision d'origine de 6 ...

Emmanuel:

Pour la classe Enceinte. Nous l'avons pas encore jointe au programme et l'avons juste conçu pour l'instant sur un fichier a part et attendons d'en savoir plus sur son utilisation pour
l'incorporée de facon idéal dans un fichier du programme.

Semaine 6 :
A partir d'ici nous avons décidé de séparer nos parties telles que: Emmanuel s'occupe de la parti graphique, des classes textViewer dessinable etc... tout ce qui touche à la "visualisation" et Justin du coté particules système choc etc...

Nous n'avons pas beaucoup avancé cette semaine mais il y a une semaine de vacance la semaine prochaine

Semaine 7 :
Emmanuel

Nous avons de la difficulté à implémenter la classe dessinable dans le programme pour moi elle créé une dépendance cyclique entre les classes héritantes de Dessinable qui ont besoin de
SupportADessin dans la définitions de Dessine_sur et SupportADessin qui utilise les sousclasses de Dessinable dans ses méthodes Dessine.
Ce problème a été résolu avec l'aide d'un assistant il suffisait de declarer sans initialiser les classes dessinables dans Supportadessin.h.Pour dire au compilateur qu'elles existent et que nous les initialiserons plus tard.
Pour l'instant nous ne savons pas encore ou ranger les classes SupportADessin et Textviewer qui sont encore dans les fichier ExerciceP7 et Systeme1
Nous arrivons à l'affichage recherché dans l'énoncé en utilisant les méthodes d'une instance de Textviewer cependant pas encore à travers la méthode dessine_sur comme souhaité.

Justin

Nous faisons attention à l'organisation et à la lisibilité de notre programme et avons décidé de réunir la classe Particule et toutes ses sous classes dans le fichier
Particule.
Par ailleur pour éviter le copier-collé on a redevint les méthodes affiche des sousclasse de Particule en utilisant la méthode hérité de Particule.
Pour la classe système et enceinte que nous avons mis dans le même .h nous avons décidé d'utiliser une vector de unique ptr sur particule pour éviter les fuites de mémoires et puis car une copie de particule n'aurait pas vraiment de sens
Je me suis ensuite attaqué à la plus grosse difficulté selon moi depuis le début du projet qui était l'évolution du système comprenant les chocs et déplacements
Apres de l'aide de notre assistant j'ai compris comment calculer les chocs contre les parois ce qui m'avait paru très flou en lisant sur le site. Pour les chocs entre particules j'ai décidé de faire des vector d'index étant donné qu'on utilise des unique par je ne peux pas passer de copie. J'ai aussi rajouté un attribut forçage pour déterminer si le premier choc était forcé comme demandé.

Semaine 8 :
Serve noté, le week end nous avons amélioré les fichiers.

Semaine 9 :
Emmanuel : J'ai suivi le tutoriel sur l'interface graphique Qt que j'ai toujours difficile à prendre en main dans un premier temps. J'ai fait l'exercice P10 en essayant de faire correspondre les classes du tutoriel 05 à celles de notre programme.

L'exercicep11 m'a pris beaucoup de temps entre le dessin des particules et de l'enceinte.
Je voulais faire de base une enceinte vraiment transparente en modifiant la composante d'opacité des couleurs j'ai réussi à le faire mais on ne voyais pas les autres côtés de l'enceinte derrière celui au premier plan.
J'ai ensuite remarqué qu'il était plus esthétique d'avoir une enceinte qui n'a pas vraiment de parroi dessiné mais que ces arrêtes. On vois beaucoup mieu la simulation maintenant.

Semaine 10 :
Justin : implémentation de l'ajout de plusieurs particules aléatoirement dans l'enceinte. Pour se faire j'ai fait une méthode de système qui appel des méthode de Neon Argon ou Neon selon le gaz qu'on veut ajouté mais j'ai du refaire une méthode dans système appelée par celle dans les gazs pour pouvoir utiliser le générateur aléatoire, sinon j'aurais pu passer l'attribut tirage par référence et include Générateur aléatoire dans les sous classes de particule mais ça aurait fait du copier coller

Semaine 11 :
Ascension, on a pas beaucoup touché au projet cette semaine

Semaine 12 :
Mouvement Brownien : nous avons décidé d'utiliser des deque pour pouvoir push front meme si des queue suffisait, comme ça la trace est éphémère ce qui est plus jolie et ne rempli pas l'enceinte au bout de 20sec. En revanche on a eu pas mal de problèmes car la trace bougeait avec la particule: elle faisait des translations qui faisait que ce n'était pas du tout ce que l'on attendait
DONNE LA SOLUTION

Pour l'autre modèle de collision il a fallu savoir comment on allait découper l'enceinte, nous avons décidé de créer un attribut vector tridimensionnel dans l'enceinte qui permettait que chaque case appartienne à l'enceinte. De plus comme on veut que l'enceinte soit partagé en case lors de son initialisation on utilise une méthode redimensionne qui fixe la taille des vestons. Je me suis aussi rendu compte que les longueurs de Enceinte devaient être positives donc j'ai rajouté un test dans le constructeur qui si échoue fixe la longueur arbitrairement à 20
Ensuite il a fallu redéfinir deux méthodes évolue : une dans système et une dans particule mais l'avantage est qu'elles ont pas mal de points commun comme le déplacement et les chocs contres les parois. Il faut donc s'occuper des déplacements des index de particules d'une case à l'autre pendant une évolution. On procède de la même manière : si plus de deux particules sont dans une même case alors on fait un tirage aléatoire.
Pour supprimer l'index dans l'ancienne case j'ai pas mal galéré car la suppression se passe dans enceinte donc lorsque j'utilisé l'iterateur pour find l'index dans la case correspondante je passais cet intérateur par copie en paramètre et donc il y avait des segmentation fault lors de l'utilisation de Erase avec l'itérateur car je n'avais pas compris que c'était un pointeur, il fallait donc le passer par référence. J'ai pallié à ce problème en faisant la recherche de l'index (find) dans la méthode de suppression de système directement

Semaine 13 :
Emmanuel :
La première extension que j'ai faite est celle qui ouvre une autre fenetre et affiche des données sur le système. L'idée d'un second glwidget mais venue directement mais je voulais le faire apparaître lors d'un evenement clavier c'est sur internet que j'ai trouvé comment l'utilisation d'un signal pouvait être interessante j'ai ensuite adapté le main pour que l'ouverture/fermeture du widget se fasse bien lors du signal.
La deuxième extension:couleur des particules selon la vitesse était moins longue que la première il suffisait juste d'ajouter les bonnes méthode et le booléen dans système et la même chose pour le changement de pas de temps.

Javais beaucoup d'idée pour les extensions, ma principale était de pouvoir sélectionnée une particule à l'aide d'un clique droit et afficher sa trace et des informations dessus. J'ai passé beaucoup de temps dessus avant de finalement abandonné car trop compliqué techniquement. Je me suis finalement résolu a un événement clavier affiche la trace de la première particules on peut bien sur l'afficher des le début sur une particule précise avec le constructeur qu'on a mis en place.

Semaine 14 :
Nous avons fait le fichier conception et README et finalisé les extensions
Et nous avons repassé en revu tous les fichiers et supprimé des erreurs telles que des includes qui trainaient dans les .cc depuis le début