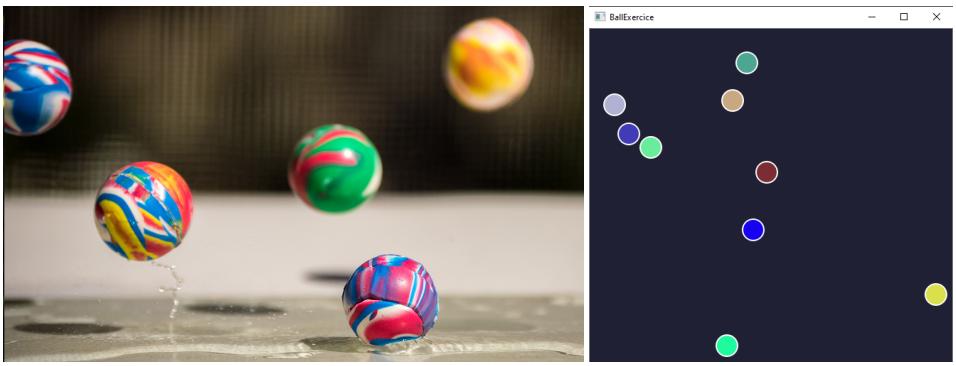
IMAC
INGÉNIEUR
ESIPE

Programmation et Algorithmique I

Projet

Attrapez les tous

Vous arrivez vers la fin des tps d'Algorithme et Programmation, vous avez obtenu les notions de bases de la programmation C++. Il est désormais temps de mettre en application les différentes notions que vous avez vues dans un même programme. Ce programme permettra à un utilisateur de jouer à petit de jeu en 2D avec des balles.



Vous allez donc devoir réaliser un programme en C/C++, le but étant :

- D'appliquer les concepts de bases de programmation (structures conditionnelles, boucles, opérations arithmétiques et booléennes)
- De vous familiariser avec le concept de structure
- De découvrir et appréhender la programmation de jeu
- Organiser votre code afin de travailler aisément sur un programme complexe

Évaluateurs :
Steeve Vincent : steeve.v91@gmail.com
Enguerrand De Smet : desmet.enguerrand@gmail.com

Modalités de rendu

- **Nombre de personnes par projet :** 2
- **Livrable :**
  - Les sources `C/C++` avec un système de compilation permettant de compiler sur un système (Linux ou Windows) avec l'environnement suivant :
    - Package : package SDL 1 et 2, SDL_image, SDL_mixer, SDL_TTF
    - Compilateur : gcc, g++ (version `C++`≤ 17)
    - Utilitaire de compilation : make, ninja, cmake
  - Un rapport au format PDF (maximum 3 pages) décrivant le projet (en particulier les aspects qui diffèrent du sujet initial, inutile de répéter le sujet), votre méthode de travail, des détails techniques si vous souhaitez détailler une fonctionnalité, les difficultés rencontrées et enfin les améliorations possibles.
  - Une vidéo ou un gif présentant succinctement la compilation et l'exécution du programme, vous n'avez pas besoin de terminer une partie, tester seulement les fonctionnalités principales. Vous pouvez utiliser Screen2Gif pour l'enregistrement.
- **Date de rendu :** Lundi 30 Janvier 2022 avant 1h00, **Tout retard sera pénalisé par des malus**

1 Jeu

Le principe est très simple, le jeu doit afficher des balles qui rebondissent sur les bords de la fenêtre, elles disparaissent quand on clique dessus, l'utilisateur doit attraper toutes les balles pour gagner.

1.1 Fonctionnalité

Cette section décrit un cahier des charges, autrement-dit, tous les éléments qui sont attendus dans le produit final. Vous verrez le détails du développement dans la section d'après.

Milestone 1 (Afficher une balle)

- Faire une structure décrivant une ellipse
- Afficher une ellipse pleine dans la fenêtre

Milestone 2 (Faire bouger la balles)

- Faire bouger la balle avec un déplacement linéaire avec un vecteur directeur initial défini aléatoirement
- Changer la direction de la balle quand elle touche les bords de la fenêtre (Inverser la vitesse en x ou celle en y selon le bord)

Milestone 3 (Gérer une liste)

- Avoir une liste chainée d'ellipses, les afficher et gérer leurs déplacements

Milestone 4 (Faire disparaitre les balles au clic)

- Faire disparaitre une balle quand on clique dessus

Milestone 4.5 (Faire apparaître des balles au clic)

- Si l'utilisateur ne clique pas sur une balle, ajouter une balle à l'endroit cliqué en lui définissant une vitesse aléatoire

Milestone 5 (Rendre le jeu personnalisable)

- Si l'utilisateur exécute le jeu avec un argument en ligne de commande, changer le nombre de balles en fonction de l'argument.
- Si l'utilisateur exécute le jeu avec plusieurs arguments, ajouter une balle par argument, l'argument définira la couleur de balle.

Milestone 6 (Mettre des murs supplémentaire)

- Gérer un tableau de murs verticaux et horizontaux, les balles devront rebondir sur ces murs

Fonctionnalités additionnelles

- Faire rebondir les balles entre-elles
- Mettre plusieurs niveaux
- Mettre une gui (afficher le nombre de balle par exemple)
- Personnaliser la couleur des balles en fonction des arguments passés en ligne de commande
- Utiliser des textures pour les balles (plaquer des images sur les ellipses)
- Faire des murs penchés (voir la section Collisions 2D)

2 Développement

Pour faire l'application vous devez respecter les contraintes décrites dans cette partie, tout écart devra être au préalable validé par vos évaluateurs et justifié dans le rapport et la soutenance.

2.1 SDL

Pour faire ce projet, il est recommandé d'utiliser la base fourni par les évaluateurs. Cette base utilise la librairie [1] SDL2 qui permet de gérer des fenêtres et leurs contenus indépendamment de la plateforme. Dans cette base vous allez retrouver aussi un main qui gère l'affichage de la fenêtre ainsi que la ***game loop***.

2.1.1 Game Loop

La plupart des jeux qui affiche continuellement des images animés (des "***frames***") font tourner une boucle de jeu qui va gérer quatre points principaux :

- gérer les inputs utilisateur
- Gérer les différents acteurs du jeu
- dessiner une frame
- petite pause

Inputs : On interroge d'abord le système pour savoir si l'utilisateur ou le système lui même ont déclenché des événements. Si oui, on récupère ces événements, on les traites un par un et on met à jour le jeu en conséquence. Pour ce jeu, nous allons notamment traiter le clique de la souris pour savoir si on clique sur une balle et l'événement de sortie (bouton fermeture, alt+f4, ...) pour quitter le jeu (pour quitter la boucle)

Gérer les Acteurs : On mets à jours l'état des acteurs et on gère leurs interactions entre-eux. Pour ce jeu, nous allons mettre à jour la position des balles et gérer la collision avec mur (voire la collision entre les balles).

Dessin : On efface le contenu de la fenêtre et on change les pixels pour représenter notre jeu dans son état actuel. Pour ce jeu, on va dessiner les balles selon leurs positions.

Pause : On fait une pause très légère pour gérer la fréquence d'affichage (***frame rate***) pour éviter de dessiner des frames qui dépassent la vitesse d'affichage de l'écran (normalement calibré sur notre capacité de vision). Pour limiter à 30 ou 60 frames par se-

1. Une librairie est un ensemble de code utilitaire pré-compilé

condes, on va dormir $\frac{1}{30}$ ou $\frac{1}{60}$ secondes. (je simplifie, normalement il faudrait prendre le temps écoulé entre deux frames).

Vous pouvez lire cet article pour avoir plus de détails sur ce principe
https://www.gameprogrammingpatterns.com/game-loop.html.
vous pouvez aussi trouver des tutoriels pour faire des jeux avec SDL dans les internets, par exemple http://www.sdltutorials.com.

2.1.2 Installation

Dans le template de code fourni, vous avez deux scripts d'installation de SDL, un pour Windows et un pour Linux. Il vous suffit d'exécuter le script pour installer SDL.

2.1.3 Votre partie

Ajoutez vos propres fichiers et modifiez le main pour faire votre jeu.

Event :
Modifiez la fonction handleEvent pour gérer le clique utilisateur. Pour ça, utilisez la structure SDL_Event fournie par la librairie. Vous pouvez accéder aux coordonnées avec l'attribut $button$ qui est du type SDL_MouseButtonEvent. Pour comprendre comment gérer les événements avec SDL, référencez vous à la documentation de SDL_Event.

Affichage :
Modifiez la fonction draw pour modifier le contenu de la fenêtre. Pour ça, utilisez la fonction $filledEllipseRGBA$ définis dans les fichiers SDL2_gfxPrimitives.c/.h. Elle prend en paramètre un SDL_Renderer (structure SDL représentant le contenu de la fenêtre), des coordonnées x et y, une largeur, une hauteur, et des composantes de couleurs rouge, vert, blue et alpha (transparence) [2].

2. Les composantes de couleurs sont des valeurs allant de 0 à 255, wikipedia:RGBA

2.1.4 Structures

Pour faire le jeu, il faudra créer et utiliser au moins les deux structures suivantes :

Ellipse :
Faites une structure qui décrit une balle :

- coordonnée X : entier
- coordonnée Y : entier
- vitesse en X : entier
- vitesse en Y : entier
- Rayon de la balle : entier
- composante Rouge : entier non-signé
- composante Vert : entier non-signé
- composante Bleue : entier non-signé

Chaine d'Ellipse :
Faites une structure permet de gérer plusieurs Ellipses avec des noeuds chainées :

- contenu du noeud : Ellipse
- noeud suivant : Pointeur vers un Noeud

Mur :
Faites une structure permet de représenter un mur :

- coordonnée du premier point X1 : entier
- coordonnée du premier point Y1 : entier
- coordonnée du deuxième point X2 : entier
- coordonnée du deuxième point Y2 : entier

Vous pouvez faire une structure **Point** pour décrire des coordonnées.

2.1.5 Fichiers/Fonctions

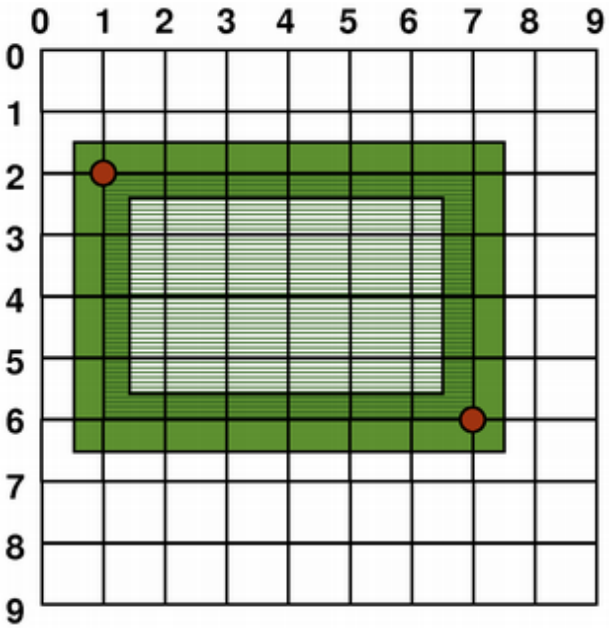
Séparer votre codes en différentes fonctions et fichiers. Implémenter notamment des fonctions pour traiter et modifier les variables de type Ellipse et Mur.

2.2 Physiques

La physique de ce jeu va être assez simple, vous allez gérer deux aspects, la vitesse et la collision

2.2.1 Coordonnées

Les coordonnées sont un peu particulier quand on dessine des éléments dans une fenêtre, l'origine se trouve dans le coin en haut à gauche de la fenêtre et l'axe y est inversé.



Du coup, lorsque vous utilisez $filledEllipseRGBA$ avec les coordonnées $(0, 0)$, vous allez dessiner une ellipse centrée dans le coin en haut à gauche.

2.2.2 Vitesse

Pour faire déplacer un élément, définissez sa vitesse, et à chaque frame (chaque tour de boucle) modifiez la position en fonction de la vitesse.

2.2.3 Collision 2D

Le système de collisions de ce jeu sera simplifié. Le jeu ne mettant en scene que des ellipses régulières, il suffit juste de prendre en compte le coordonnées de l'ellipse et de son rayon pour déterminer quand est-ce qu'elle touche un bord. Par exemple, Si vous testez le bord droit, regardez la coordonnées x de l'ellipse en retranchant son rayon et si le résultat est négatif ou nul, alors l'ellipse touche le bord droit.
Selon le bord, inversez une des composante de vitesse au moment de la collision.
Pour ceux qui veulent aller plus loin, n'hésitez pas à suivre le tuto développez

3 Notation

La note du projet sera prise en compte dans la note final de la matière Programmation et Algorithmique.

Rapport ⋆

Soutenance ⋆⋆

Compilable par l'évaluateur ⋆⋆

Propreté du code ⋆

Fonctionnalités du jeu ⋆ ⋆ ⋆⋆

Instructions **if, for, while**... ⋆⋆

Fonctions ⋆ ⋆ ⋆

Pointeurs ⋆ ⋆ ⋆

Liste chainée ⋆ ⋆ ⋆⋆

Structures ⋆ ⋆ ⋆⋆

Bonus (2) (Soutenance++, Fonctionnalités ++, Corruption, ...)

4 Remarques

— Il est très important que vous réfléchissiez avant de commencer à coder aux principaux modules, algorithmes et aux principales structures de données que vous utiliserez pour votre application . Il faut également que vous vous répartissiez le travail et que vous déterminiez les tâches à réaliser en priorité.

— Vous pouvez utiliser un outil de répartition et suivi de tâches, par exemple Trello, Azendoo ou Notion

— Ne rédigez pas le rapport à la dernière minute sinon il sera bâclé et cela se sent toujours.

— N'oubliez pas de tester votre application à chaque spécification implémentée. Il est impensable de tout coder puis de tout vérifier après. Ne visez pas directement votre oeuvre finale, tester les implémentations basiques sur une scène simple.

— Vos chargés de TD et CM sont là pour vous aider. Si vous avez des doutes/des difficultés sur un point, n'attendez pas la soutenance pour nous en parler.