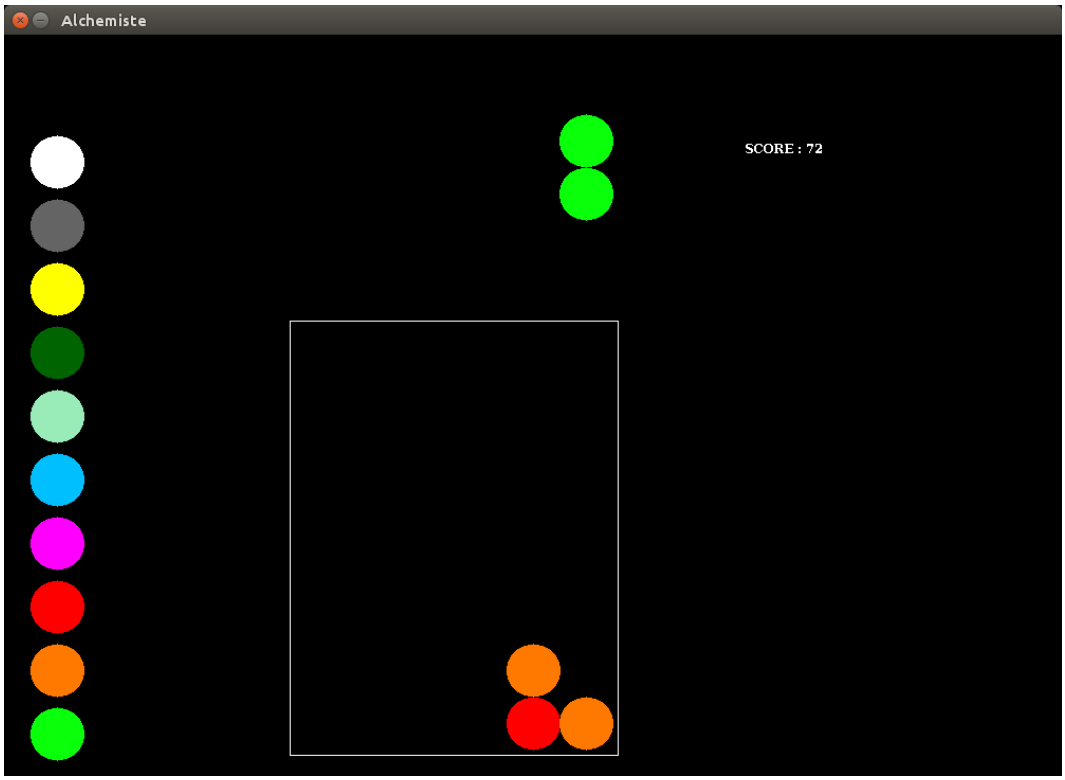
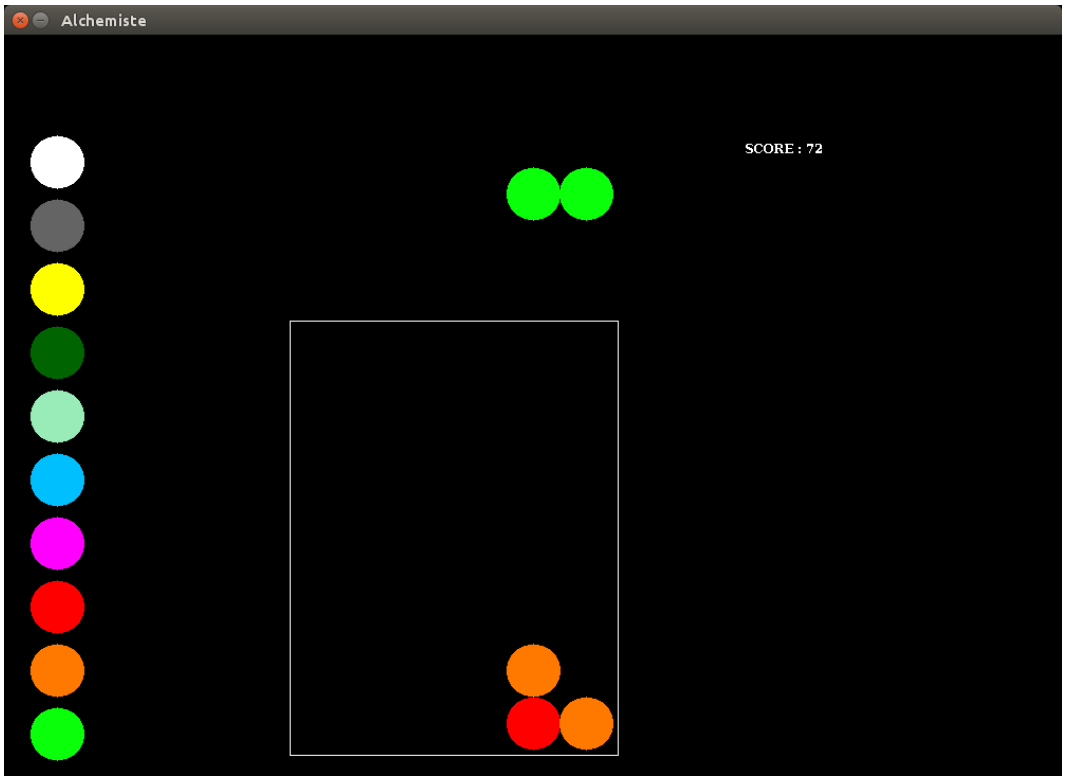
Programmation C avancée
TP 4

L'objectif de ce TP est de programmer un petit jeu "tetris-like" avec interface graphique en utilisant la LibMLV : un alchimiste.

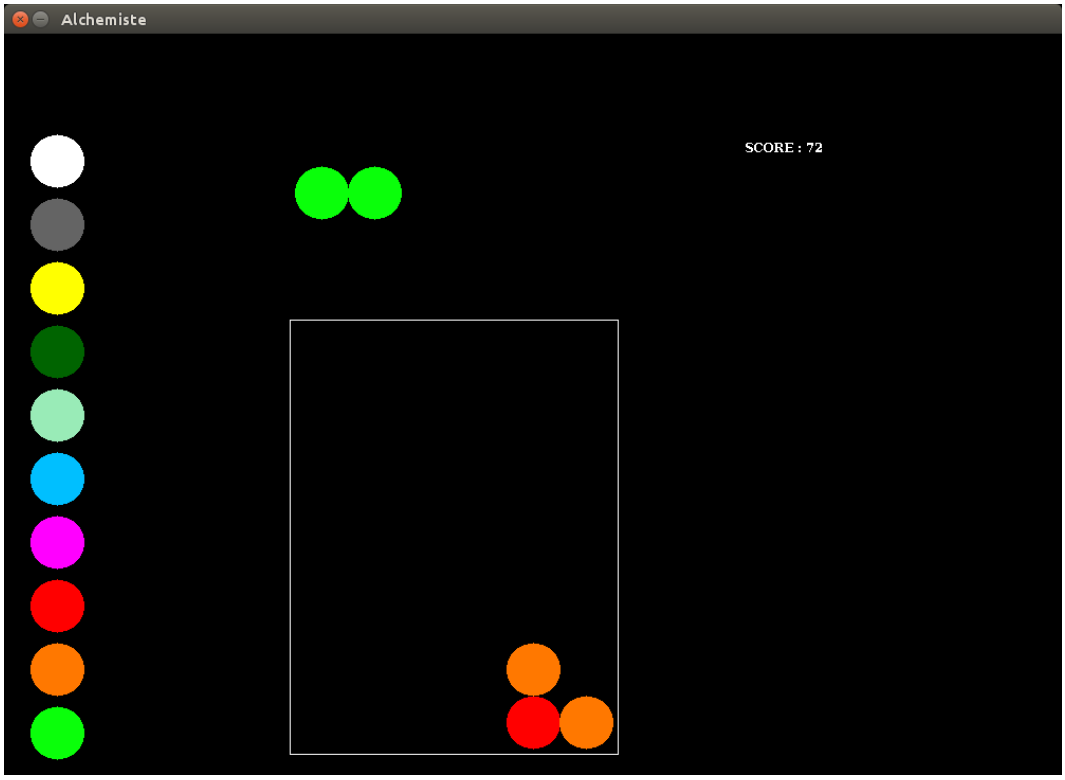
Voici un premier aperçu du plateau de jeu. On y voit un récapitulatif des objets pouvant tomber à gauche qui sont ici des boules de différentes couleurs. Au milieu se trouve la grille dont seul le contour est ici dessiné. Cette grille peut ici contenir 6 boules en lignes et 8 en colonnes. Au dessus de la grille se trouve les deux prochaines boules que l'utilisateur peut laisser tomber. Enfin à droite, il y a l'affichage du score courant de la partie.



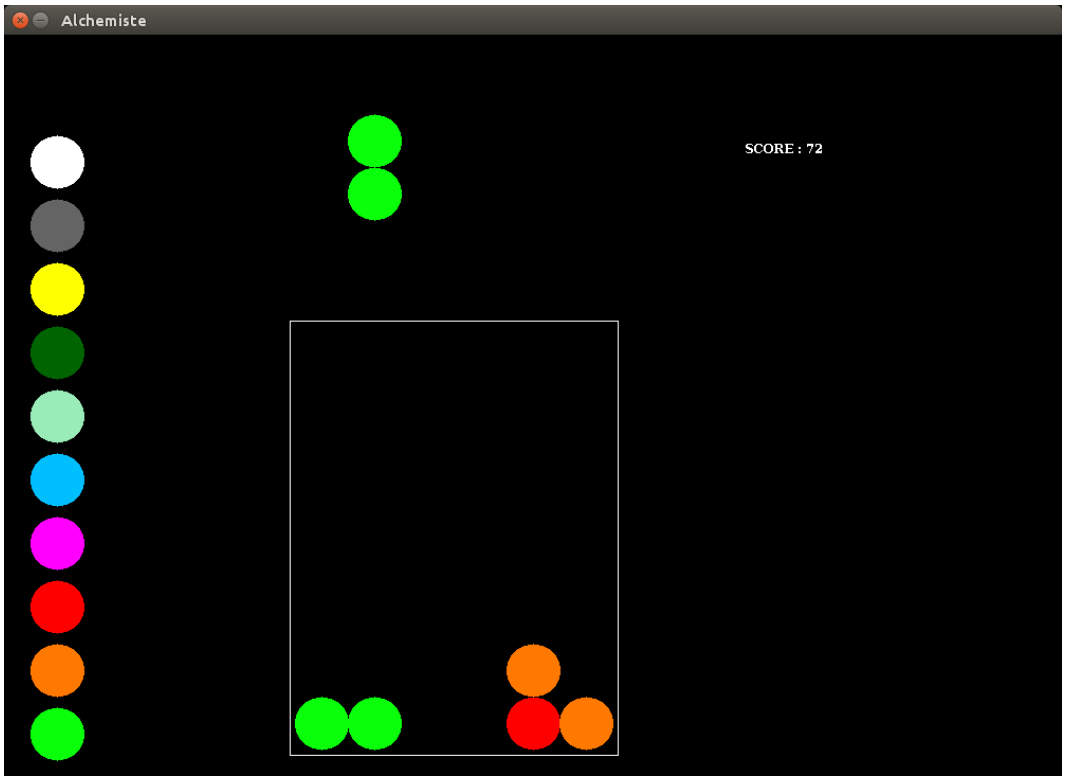
A chaque tour, l'utilisateur peut faire descendre 2 boules dans le plateau. Les boules peuvent tomber verticalement comme dans la première image ou horizontalement comme l'image qui suit. Pour effectuer la rotation des deux boules, on peut enfoncer la flèche du haut sur le clavier. La flèche du bas laisse tomber les deux boules.



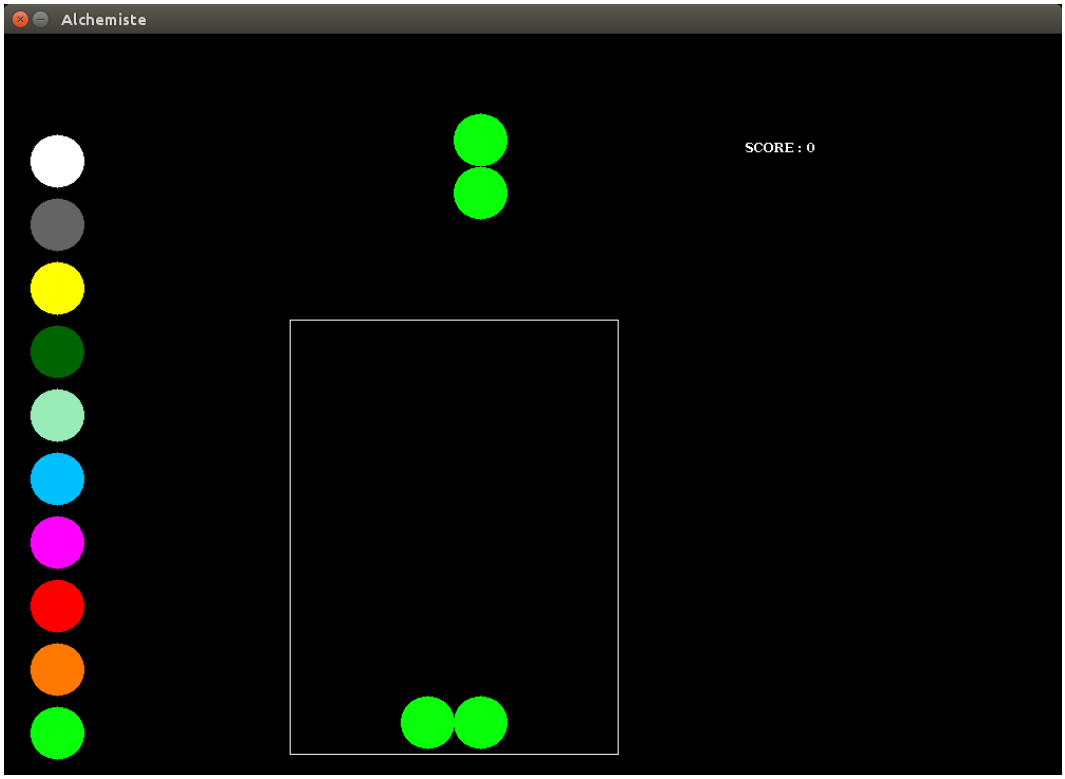
On peut promener les deux boules à gauche ou à droite sans sortir des 6 colonnes.



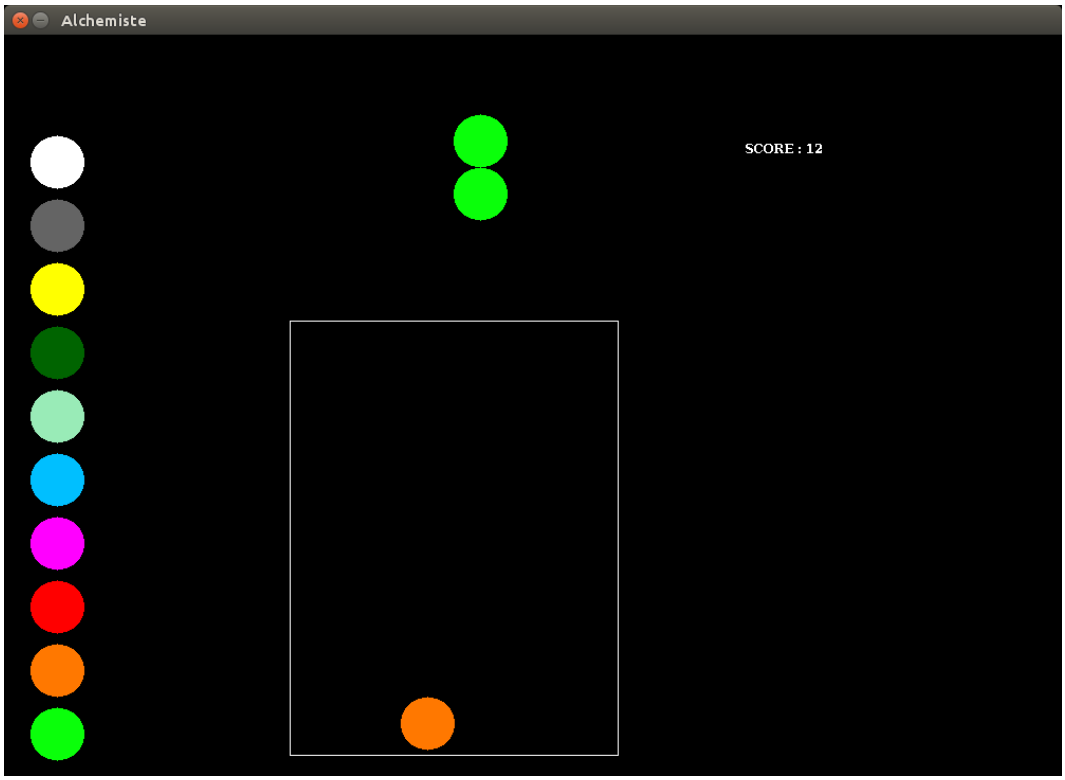
Tant que l'utilisateur ne fait pas tomber les boules (avec la flèche du bas), on peut les remettre à la verticale à souhait ou encore les promener.



Lorsqu'au moins 3 boules de même couleur se touchent (via gauche, haut, droite ou bas... Les boules diagonales ne se touchent pas !), le paquet correspondant explose pour se transformer en une boule de poids supérieur (on monte sur l'échelle des couleurs). Sur le dessin suivant, deux boules vertes vont bientôt toucher deux boules vertes qui se touchent déjà.



En faisant tomber ces deux boules avec la flèche du bas, le paquet de 4 boules vertes disparaît alors et laisse la place à une boule de poids supérieur (ici l'orange) qui se placera le plus en bas et le plus à gauche de la zone contiguë formée par les boules qui explosent.



La zone libérée laisse peut-être des boules en suspension (il n'y a rien dessous). Vous devez faire tomber les boules restantes par gravité et relancer votre mécanisme de détection des boules voisines. Lorsque plusieurs explosions en chaînes ont lieu lorsque l'utilisateur fait tomber les bonnes boules au bon endroit, on parle de combo. Ici l'utilisateur marque 12 points.

Le game over arrive lorsque des boules devraient sortir du plateau de jeu. Le dernier score est le score du joueur à sa partie.

A chaque explosion, le joueur gagne des points. Les points marqués correspondent à la somme des boules disparues pondérées par leur poids. Dans le prototype de votre enseignants, le vert vaut 3 points, $3^2 = 9$ points pour l'orange, $3^3 = 27$ points pour le rouge, $3^4 = 81$ points pour une boule violette et ainsi de suite.

Les boules à faire tomber ont des couleurs aléatoires parmi les boules que l'utilisateur a réussi à faire apparaître en combinant les boules, sauf la dernière. Si l'utilisateur a fabriqué une boule violette au plus, les deux prochaines boules seront tirées parmi le vert avec probabilité $\frac{1}{3}$, l'orange avec probabilité $\frac{1}{3}$ et enfin le rouge avec probabilité $\frac{1}{3}$. En début de partie, il n'y a pas de condition pour le vert (quand le plateau est vide, il faut tout de même bien autoriser une couleur...).

Modélisation du problème :

Vous avez carte blanche pour modéliser le plateau de jeu. Juste pour information, voici les structures que votre enseignant a utilisées pour réaliser un prototype. N'hésitez pas à changer ces choses si vous pensez faire mieux.

```
1 typedef struct position{
2   int alignement;
3   int vertical;
4   int left_level;
```

```
5     int right_level;
6   }Position;
7
8   typedef struct plateau{
9     int cases[8][6];
10    Position pos;
11  }Plateau;
12
13  typedef struct cellule{
14    int index;
15    struct cellule* next;
16  }Cellule;
17
18  typedef Cellule* Voisinage;
```

La structure position détermine la position et les couleurs des deux boules s'apprêtant à tomber. La structure Plateau modélise le jeu dans un certain état à l'instant t. Des listes chaînées d'entiers ont ensuite été utilisées pour déterminer les zones contenant des boules de la même couleur et adjacentes. Dans la structure plateau, la grille est codée via un tableau à deux dimensions d'entier. 0 signifie que l'emplacement correspondant est vide, 1 code une boule verte, 2 pour une boule orange, 3 pour une boule rouge et ainsi de suite.

Pour une première critique, il aurait été plus judicieux d'utiliser un autre tableau d'entier pour la détection des boules de même couleur adjacentes. La fainéantise de votre enseignant l'a poussé à ne pas recoder cela proprement.

Suggestions d'amélioration : (optionnel, pour ceux qui trouvent les TP trop simples...)

- Changer les cercles de couleurs par des images, des personnages ou tout autre chose qui vous paraîtrait marrant. Des dessins d'animaux formant une chaîne alimentaire est une idée.
- Faire un tableau récapitulatif des high scores.
- Montrez quelles sont les prochaines boules qui apparaîtront au tour prochain (laisse à l'utilisateur le choix d'organiser un peu de stratégie).
- Faire des animations de destruction des boules et de la chute par gravité des éléments restant sur le plateau. C'est pas forcement facile mais c'est drôle ! La dernière version de la libMLV supporte des animations frame par frame (à vérifier...).
- Mettre un compteur pour enregistrer le temps de jeu.
- Tout ce qui vous parait sympathique à rajouter...

La LibMLV est la bibliothèque de l'université pour les petits projets graphiques en langage C. Cette bibliothèque interface la SDL et ainsi emballe des fonctions difficiles à utiliser dans des fonctions claires et simples. Cette dernière est disponible sous système Unix ou windows pour des systèmes 32 ou 64 bits. Le projet est disponible à la page :
`http://www-igm.univ-mlv.fr/~boussica/mlv/api/French/html/index.html`
La documentation des fonctions a été automatiquement générée par `doxygen` et cette dernière est disponible en français à l'adresse :

`http://www-igm.univ-mlv.fr/~boussica/mlv/api/French/html/globals_func.html`

Un module utilisant la LibMLV doit inclure les entêtes de la LibMLV :

```
1 #include <MLV/MLV_all.h>
```

Si vos fonctions n'utilisent pas de type définis dans la LibMLV, vous pouvez confiner cette inclusion dans un fichier de code `.c`, sinon il faudra le mettre dans les headers du module.

Les modules important la LibMLV doivent être compilés avec le flag `-lMLV`.

Voici la liste des fonctions issues de la LibMLV utilisées par votre enseignant pour réaliser un prototype de votre TP :

- MLV_create_window (création d'une fenêtre)
- MLV_free_window (libération de la fenêtre)
- MLV_actualise_window (demande d'opérer les changements d'affichage à l'écran)
- MLV_draw_text (afficher un texte)
- MLV_draw_filled_rectangle (pour tout effacer, on met un rectangle noir dans la fenêtre)
- MLV_convert_rgba_to_color (fabrication d'une couleur)
- MLV_get_random_integer (fabriquer un entier aléatoire)
- MLV_get_event (récupération des mouvements et clics de la souris)
- MLV_wait_seconds (attendre quelques secondes une fois le jeu fini)
- MLV_wait_keyboard (attend que l'utilisateur enfonce une touche du clavier)

Dernières consignes et remarques :

- Faites du code de qualité (documentation).
- Faites un compte rendu pour ce TP4.
- Modularisez agréablement votre code.
- Faites de la compilation séparée avec des jolis `makefile`.