Clément
GRENOT
4A-IT-ILC

19/01/22

Compte rendu n°5 :

Projet 4A – ILC :
Développement Pac-Man en C++

Ajout des fonctionnalités de déplacement Pac-Man et de déplacement automatique des fantômes.

```
void BaseCharacter::GoUp() {
    if (PosL > 0 && Matrix[PosC][PosL - 1] != '0') {
        PosL--;
    }
}
void BaseCharacter::GoDown() {
    if (PosL < NbL - 1 && Matrix[PosC][PosL + 1] != '0')
        PosL++;
}
void BaseCharacter::GoLeft() {
    if (PosC > 0 && Matrix[PosC - 1][PosL] != '0') {
        PosC--;
    }
}
void BaseCharacter::GoRight() {
    if (PosC < NbC - 1 && Matrix[PosC + 1][PosL] != '0') {
        PosC++;
    }
}
```

Ajout dans la classe 'BaseCharacter'

Cet ajout permet faire se déplacer les personnages sous la condition qu'ils n'essayent pas de rentrer dans un mur.

Pour que les fantômes se déplacent de manière autonome, on ajoute un générateur de nombres aléatoire qui définira dans quelle direction ce dernier ira. On fini par une mise a jour de l'affichage pour permettre de voir les fantômes se déplacer en temps réel.

```
void Ghost::DeplacementAuto()
{
    int hasard = rand() % 4;

    switch (hasard)
    {
    case 0:
        GoUp();
        break;
    case 1:
        GoDown();
        break;
    case 2:
        GoLeft();
        break;
    case 3:
        GoRight();
    default:
        break;
    }
    glutPostRedisplay();
}
```

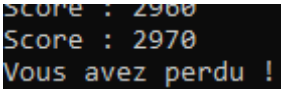
Méthode 'DéplacementAuto'

Ajout d'un test de collision entre Pac-Man et les fantômes qui conduit à un arrêt du programme et à la fin de la partie.

On effectue ce test a chaque fois que le joueur se déplace mais aussi a chaque fois qu'un fantôme se déplace. On fait cela car il était possible de passer par-dessus un fantôme sans que cela soit détecté comme une collision.

```
void Ghost::TestCollision()

{
    if (PosC == pacman.GetPosC() && PosL == pacman.GetPosL())
    {
        cout << "Vous avez perdu !" << endl;
        LabyDraw();
        system("pause");
        exit(0);
    }
}
```

Déclaration méthode 'TestCollision'



Affichage console d'un message de fin de partie

Ajout de la possibilité de téléportation du joueur via le fichier texte de définition du niveau. Le personnage principal peu ainsi passer d'un coté à l'autre du niveau simplement en se rendant sur les cases dédiées, comme dans la version originale du jeu.

```
case 'x':
    Tp1C = i;
    Tp1L = j;
    break;
case 'y':
    Tp2C = i;
    Tp2L = j;
    break;
```

Lecture des points de TP dans le fichier de niveau

Déclaration de la classe ’Point’ et donc des fichiers ‘point.h’ et ‘point.cpp’.

```h
#ifndef POINT_H
#define POINT_H

#include <iostream>
#include "Hero.h"

using namespace std;

class Point
{
private:
    Point* Next;
    int PosiC, PosiL;
public:
    // Accesseur
    Point* GetNextP() const { return Next; }
    int GetPosiC() const { return PosiC; }
    int GetPosiL() const { return PosiL; }

    // Mutateur
    void SetSuivant(Point* ptr) { Next = ptr; }
    void SetPosiC(int C) { PosiC = C; }
    void SetPosiL(int L) { PosiL = L; }

    // Constructeur
    Point() {
        PosiC = PosiL = '0';
        Next = NULL;
    }

    void Draw();
    //int Mange();
};
#endif
```

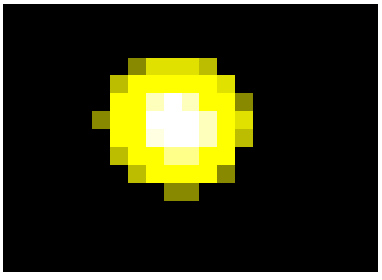
Fichier 'point.h'

Les points sont ajoutés sous forme d’itération de liste chainée, ce qui permet de facilement les supprimer lorsque le joueur passe sur un pont.

```cpp
#include "Point.h"
#include "Hero.h"

extern Hero pacman;

void Point::Draw()
{

    glPushMatrix();
    glTranslated(PosiC + 0.5, PosiL + 0.5, 0.0);    // Définition du centre de notre zone de dessin
    glColor3d(1, 1, 0);
    glutSolidSphere(0.15, 12, 12);
    glColor3d(1, 1, 1);
    glutSolidSphere(0.07, 12, 12);
    glPopMatrix();
}
```

Fichier 'point.cpp'

On défini ici le graphisme d’un point, qui sera utilisé pour tous les autres lors de l’affichage du niveau. Il s’agit simplement d’un cercle blanc dans un cercle jaune.



Ajout de l'incrémentation du score lorsque le joueur mange un point

```
while (P != NULL)
{
    if (pacman.GetPosC() == P->GetPosiC() && pacman.GetPosL() == P->GetPosiL()) {
        Save = P->GetNextP();
        P = Precedent;
        P->SetSuivant(Save);
        score = score + 10;
        cout << "Score : " << score << endl;
    }
    Precedent = P;
    P = P->GetNextP();
}
```

Lorsque le joueur passe sur un point, celui-ci est supprimé de la liste chainée et on incrément le score du joueur d'un certain montant de points.

```
Score : 2780
Score : 2790
Score : 2800
Score : 2810
Score : 2820
Score : 2830
Score : 2840
Score : 2850
Score : 2860
Score : 2870
Score : 2880
Score : 2890
Score : 2900
Score : 2910
Score : 2920
Score : 2930
Score : 2940
Score : 2950
Score : 2960
Score : 2970
```

Affichage dans la console du score