# Compte rendu : TP3 – MOTEUR DE JEUX

Vincent de Rousiers

## Résumé :

- Ajout d'un lien entre les différentes fenêtre via les classes QTcpSocket et QtcpServer.
- Ajout de slots pour gérer les connections et les actions à effectuer entre les fenêtres.
- Ajout d'une méthode d'initialisation des saisons (void TriangleWindow::setSeason(int i)) et d'une méthode d'affichage des saisons (void TriangleWindow::displaySeasons()).
- Ajout d'une classe Particules qui permet de stocker l'ensemble des points représentant les effets météorologiques.
- Les saisons sont représentées ainsi :
  - Hiver : couleur blanche au sol, chute de neige grisâtre et lumière sombre
  - Printemps : couleur vert clair au sol et lumière bleu/gris clair
  - Été : couleur jaune au sol et lumière bleu
  - Automne : couleur marron au sol, chute de pluie bleue et lumière orangée

## Partie réseau :

Deux nouveaux attributs sont ajoutés à la classe TriangleWindow afin de savoir si l'on est client ou serveur : socket et server. Le serveur stockera les clients qui se sont connectés à lui dans une liste d'adresses : std::vector<QTcpSocket*> sockets.
Le serveur va lancer un timer (timerFiveMinutes) de 5s (5min était beaucoup trop long pour tester et débugger) qui déclenchera un envoi de message à tous les clients pour leur faire changer de saison :

```
timerFiveMinutes = new QTimer();
timerFiveMinutes->connect(timerFiveMinutes,
                  SIGNAL(timeout()),this,SLOT(changeSeason()));
timerFiveMinutes->setInterval(1000*5);
timerFiveMinutes→start();



void changeSeason(){

      for(int i = 0 ; i < sockets.size() ; i++){
            sockets.at(i)->write("season");
      }
      season++;
      season %= 4;
      setSeason(season);
      //qDebug()<<"Je suis à la saison "<<season<<endl;
}
```

# Partie saisons :

## Affichage :

La gestion des saisons se fait via un nouvel attribut : *season*
Celui-ci permet de déterminer quelle saison la fenêtre doit représenter (0 pour l'hiver, 1 pour le printemps, 2 pour l'été et 3 pour l'automne).

Deux nouvelles fonctions vont elles gérer l'affichage des saisons :

- ```
  void TriangleWindow::setSeason(int i){
        season = i;
        GLfloat p[3];
        switch(season){
        case 0:
              particules = new Particules(&m_image);
              p[0] = 0.8f; p[1] = 0.8f; p[2] = 0.8f;
              particules->setParticuleColor(p);
              particules->setParticuleSize(2);
              particules->init(0.04,1,3);
              break;
        case 1:
              particules = NULL;
              break;
        case 2:
              particules = NULL;
              break;
        case 3:
              particules = new Particules(&m_image);
              p[0] = 0.0f; p[1] = 0.0f; p[2] = 0.8f;
              particules->setParticuleColor(p);
              particules->init(0.02,1,3);
              break;
        }
  }
  ```

Celle-ci va initialiser l'attribut *season* puis mettre en place les particules représentant la neige ou la pluie pour l'hiver et l'automne.

Particules est une nouvelle classe contenant un tableau de points permettant de gérer les différentes particules. Chaque point possède une position (x,y,z) ainsi qu'une vitesse de déplacement. Il disparaît une fois qu'il a touché le sol et est remplacé par un nouveau point placé aléatoirement au dessus de la map.

- ```
  void TriangleWindow::displaySeasons(){

        switch (season){

        case 0://Hiver
              glClearColor(0.5f,0.5f,0.5f,1.0f);

              particules->update();
              glColor3f(1.0f, 1.0f, 1.0f);
              break;
        case 1://Printemps
              glClearColor(0.75f,0.85f,1.0f,1.0f);
              glColor3f(0.0f, 0.75f, 0.0f);

              break;
        case 2://Ete
              glClearColor(0.0f,0.5f,0.8f,1.0f);
              glColor3f(1.0f, 1.0f, 0.0f);
  ```

```
            break;
        case 3://Automne
            glClearColor(0.92f,0.57f,0.13f,1.0f);
            particules->update();
            glColor3f(0.37f, 0.2f, 0.07f);

            break;

        default:
            break;
        }

    }
```

Cette fonction va quant à elle s'occuper d'appeler la fonction d'update de la classe Particules qui permet à la neige ou à la pluie de tomber. Elle colorie en même temps la map et l'arrière plan.

## Particules :

```
class Particules{
public:
    Particules();
    Particules(QImage*);

    //taux entre 0 et 1 pour connaitre le nombre de particules à afficher par
    //rapport aux nombres de points représentant la map
    void init(GLfloat taux, GLfloat vitesseMin, GLfloat vitesseMax);

    Point** getPoint();
    void update();

    void setParticuleSize(int);
    void setParticuleColor(GLfloat*);

private :
    Point **points; //le tableau de points
    int nbParticules; //le nombre de points/particules affichées
    QImage *image; //l'image sur laquelle tombent les particules
    GLfloat width; //tailles de l'image
    GLfloat height;

    int particuleSize; //taille des particules
    GLfloat *particuleColor; //couleur des particules
};
```

Cette classe permet la gestion des particules, chaque Point dans points est un flocon ou une goutte tombant sur la map, quand il l'a touche il disparaît et un nouveau est créé aléatoirement au dessus. On peut gérer la taille des particules ainsi que la couleur.
L'initialisation permet de donner un pourcentage de particules à créer (si la map contient dix points et que taux est à 1, dix particules seront créées, si taux est à 0.5, les particules ne seront plus que cinq) ainsi que la vitesse minimale et maximale de chute.

## Bonus :

- On pourrait donner une durée de vie aux particules qui touchent le sol, tant qu'elles restent « vivantes », si une nouvelle tombe à côte, on réinitialise la durée de vie de la première. Si on trouve un certain nombre de particules au sol et dans un espace réduit, on crée une zone d'accumulation pour représenter une rivière ou une flaque.
- Pour simuler des effets météo, on pourrait définir une zone de la map où les particules seraient soumises à du vent (déplacement sur l'axe Z et aussi aléatoirement sur les axes X et Y) ou plus grosses en hauteur (augmenter la taille des particules en haut des collines).