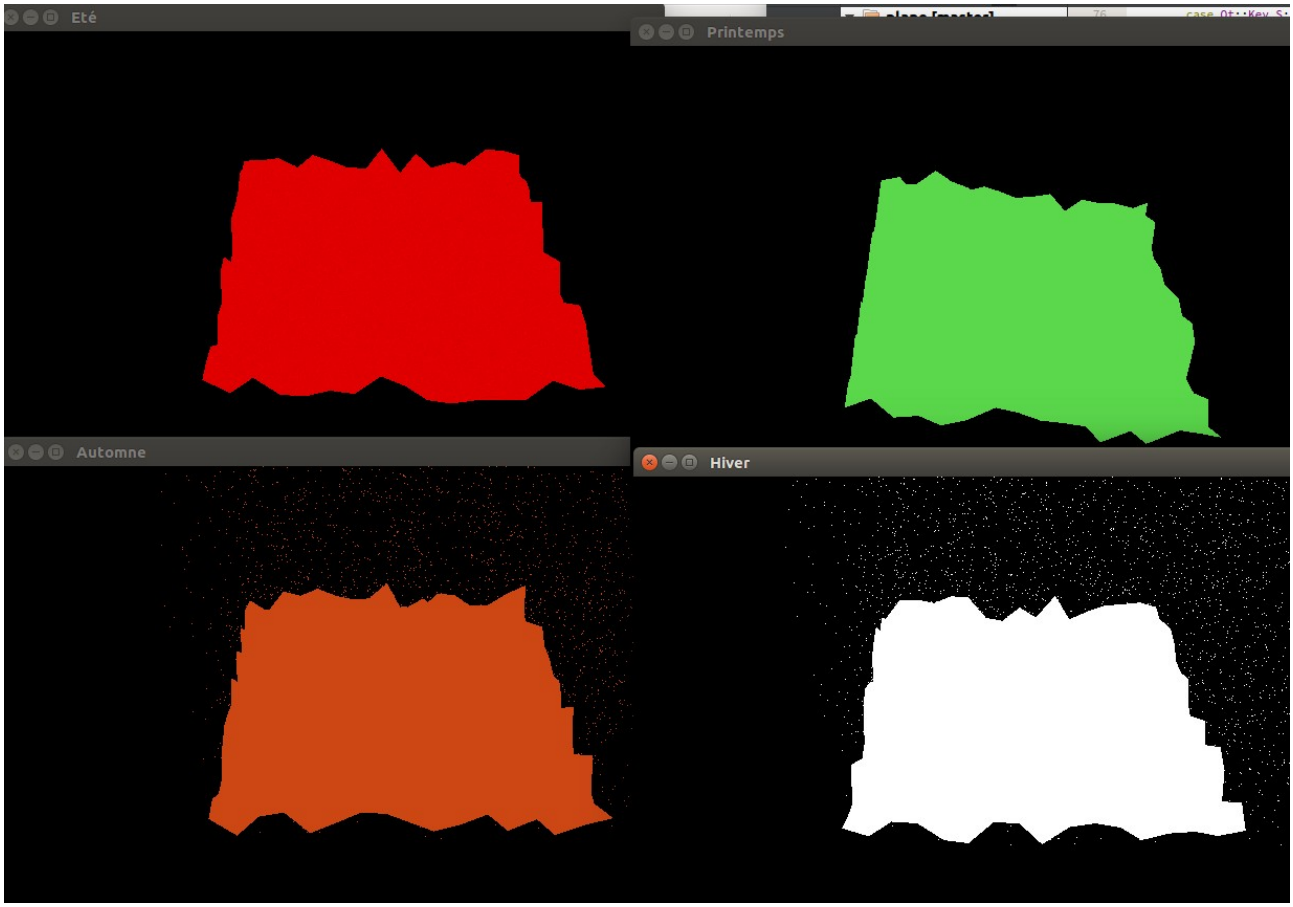
Compte Rendu TP3
Cossin Tristan

Synchronisation des fenêtres :



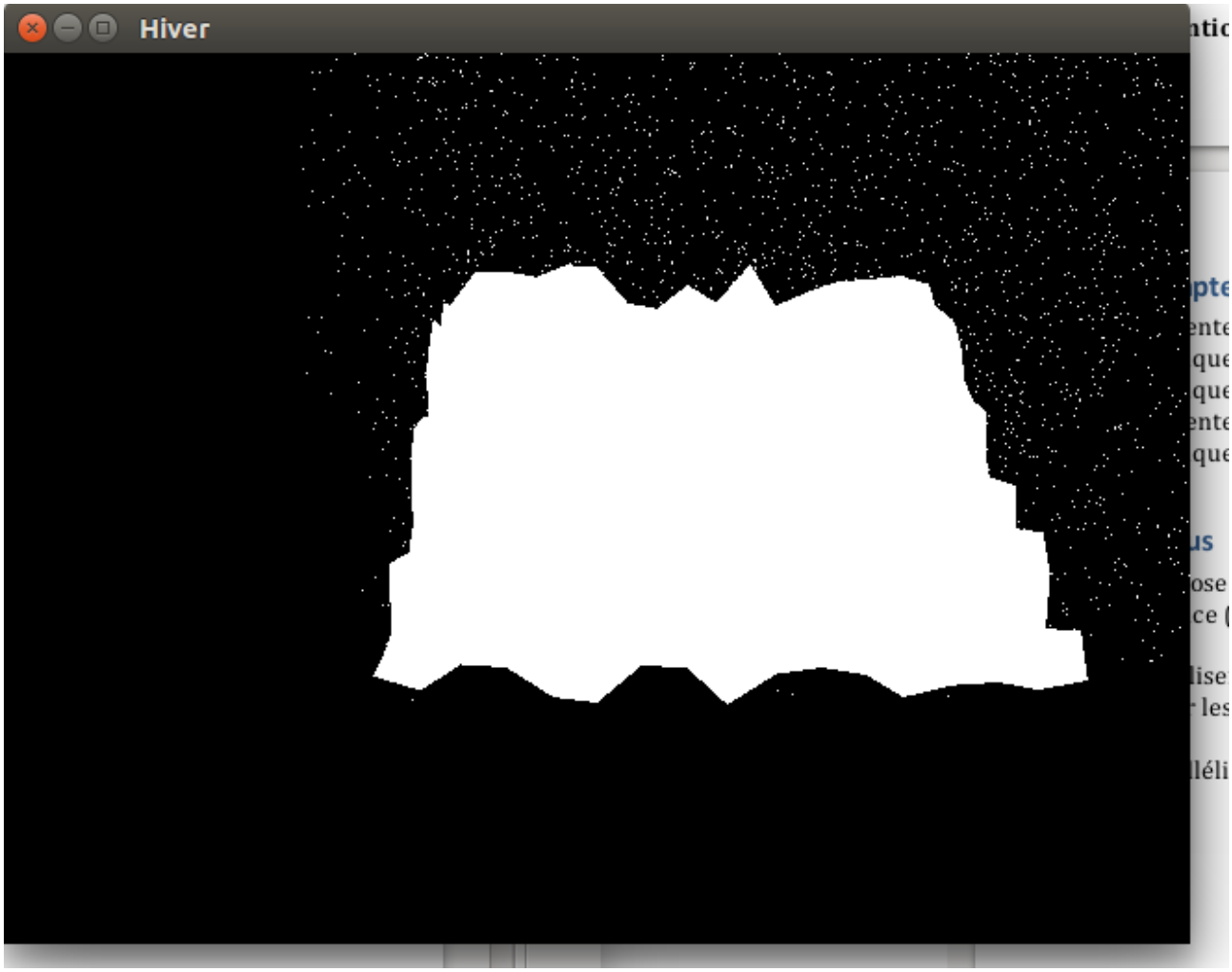
Pour réaliser la synchronisation , j'ai fais partir les fenêtres de départ à une saison différente chacune puis à chaque changement, la saison de la fenêtre avance d'une saison ce qui permet d'avoir les 4 saisons en même temps.

Pour le changement, j'ai utiliser les SLOT de Qt avec comme déclencheur un timer.

Il n'y a pas de gestion de la lumière pour l'été ou une autre saison car je n'ai pas réussi.

```
MainWidget *widget1 = new MainWidget(nullptr, 30, 1); //1 = printemps
MainWidget *widget2 = new MainWidget(nullptr, 30, 2); //2 = été
MainWidget *widget3 = new MainWidget(nullptr, 30, 3); //3 = automne
MainWidget *widget4 = new MainWidget(nullptr, 30, 4); //4 = hiver

widget1->show();
widget2->show();
widget3->show();
widget4->show();

QTimer* seasonTimer = new QTimer;

QObject::connect(seasonTimer, SIGNAL(timeout()), widget1, SLOT(SaisonSuivante()));
QObject::connect(seasonTimer, SIGNAL(timeout()), widget2, SLOT(SaisonSuivante()));
QObject::connect(seasonTimer, SIGNAL(timeout()), widget3, SLOT(SaisonSuivante()));
QObject::connect(seasonTimer, SIGNAL(timeout()), widget4, SLOT(SaisonSuivante()));

seasonTimer->start(5000);
```

Particules :



Pour la création des particules, il n'y a pas de classe ou de structure quelconque, Pour y arriver, j'ai crée un tableau dynamique de QVector3D qui enregistre la position des points (dans la classe MainWidget), un int compris entre 1 et 4 pour connaître la saison et un int pour connaître le nombre de particule à généré ou traité.

A la création, on défini aléatoirement une position X, Y et Z à notre point (point = particule) puis à chaque traitement des particules, on les fait descendre d'un pas constant. Si une particule est descendu trop bas, on la repositionne en haut .

```
switch (Saison)
{
    case 1:
        texturePrintemps->bind(0);
        program.setUniformValue("texturePrintemps", 0);
        break;

    case 2:
        textureEte->bind(0);
        program.setUniformValue("textureEte", 0);
        break;

    case 3:
        textureAutomne->bind(0);
        program.setUniformValue("textureAutomne", 0);
        break;

    case 4:
        textureHiver->bind(0);
        program.setUniformValue("textureHiver", 0);
        break;
}

// Draw cube geometry
geometries->drawPlaneGeometry(&program);

//on fais bouger les particules
if (NombreParticule > 0)
{
    glBegin(GL_POINTS);
    if (Saison == 1)
        glColor3f(0,0,1);
    else
        glColor3f(1,1,1);

    for (int i = 0; i < NombreParticule; i++)
    {
        Particules[i].setY(Particules[i].y() - 1);
        if(Particules[i].y() < 0)
        {
            Particules[i] = QVector3D(generateRand(20), generateRand(20), generateRand(256));
        }
        glVertex3f(Particules[i].x(), Particules[i].y(), Particules[i].z());
    }
    glEnd();
```

```
    }
}


void MainWidget::SaisonSuivante()
{
    Saison++;

    //delete texture;

    if(Saison == 5)
        Saison = 1;

    switch (Saison)
    {
        case 1:
            setWindowTitle("Printemps");
            if(NombreParticule != 0)
                delete[] Particules;
            NombreParticule = 0;
            break;

        case 2:
            setWindowTitle("Eté");
            NombreParticule = 0;
            break;

        case 3:
            setWindowTitle("Automne");
            if(NombreParticule == 0)
            {
                NombreParticule = 60000;
                Particules = new QVector3D[NombreParticule];
                for (int i = 0; i < NombreParticule; i++)
                {
                    Particules[i] = QVector3D(generateRand(20), generateRand(20), generateRand(256));
                }
            }
            break;

        case 4:
            setWindowTitle("Hiver");
            if(NombreParticule == 0)
            {
                NombreParticule = 60000;
                Particules = new QVector3D[NombreParticule];
                for (int i = 0; i < NombreParticule; i++)
                {
                    Particules[i] = QVector3D(generateRand(20), generateRand(20), generateRand(256));
                }
            }
            break;
    }
}
```

Bonus :

Pour accumuler les particules, on pourrait mettre en place un système de compteur sur chaque sommet de la map, et a chaque particule tombée, la particule ajoute 1 au compteur du sommet sur lequel elle est tombée. Ainsi, en fonction du nombre de particule sur le sommet ou même des sommets voisins, on peut déterminer différentes actions comme la création d’un lac ou un tas de neige.