# Travail Pratique #2

## Application graphique Paint++

présenté à

**Philippe Voyer**

par

Équipe 23

Université Laval

11 Mars 2016

# Sommaire

L'application Paint3D+ que nous développons est un programme d'édition graphique interractif dans le style de PaintdotNet. Elle comporte 2 modes : mode 2D et mode 3D.

Concrètement, l'application permet de construire des primitives vectorielles en 2D (ligne, triangle, rectangle et cercle) et d'afficher un modèle 3D pour ensuite modifier ses propriétés. Il est aussi possible d'importer des images et d'exporter le contenu de la scène en image.

Une interface intuitive est affichée lors du démarrage du programme et l'utilisateur peut interagir à l'aide des menus et des panaux graphiques. D'autres possibilités de manipulation sont aussi décrites plus loins dans le rapport.

Toutes les entités géométriques sont organisées dans une hiérarchie de classes et elles peuvent être maniupulées par différentes méthodes (on est capable d'appliquer une translation, rotation, changer l'échelle, etc.). Les primitives peuvent être composées dans une seule entité.

# Interractivités

Test

# Technologies

Test

# Architecture

Test

# Fonctionnalités

## 5.1 Image

### 5.1.1 Importation

Pour satisfaire ce critère, l'application doit pouvoir importer des fichiers images (.jpg, .png, etc.) pour les utiliser à des fins de rendu graphique.

L'application permet cette fonctionnalitée grâce à l'utilisation du bouton "Import". Celui-ci ouvre un navigateur de fichier qui permet à l'utilisateur de selectionner facilement le fichier image voulu. Cette image est placée dans le renderer 2D, initialement avec sa taille d'origine et à la position (0,0).

Les figures suivantes demontrent cette fonctionnalitée.

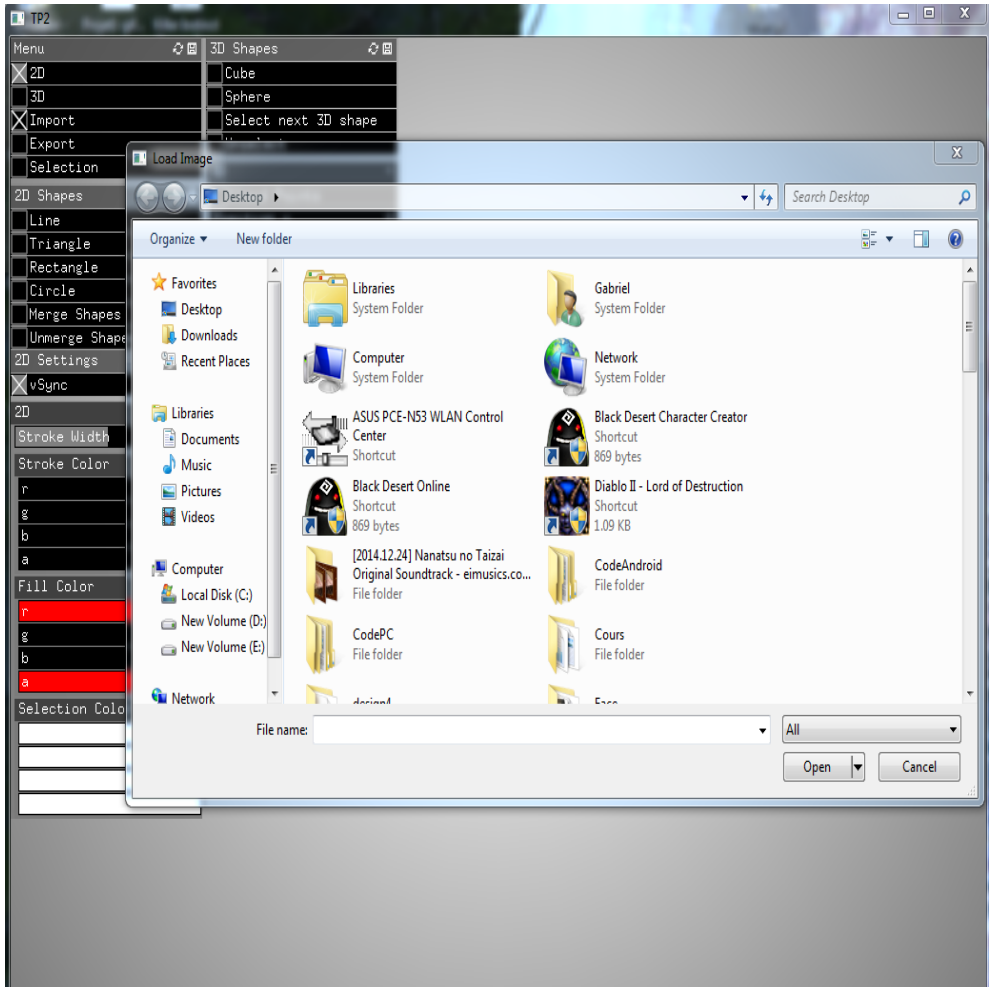


FIGURE 5.1 – Le bouton Import et son navigateur.

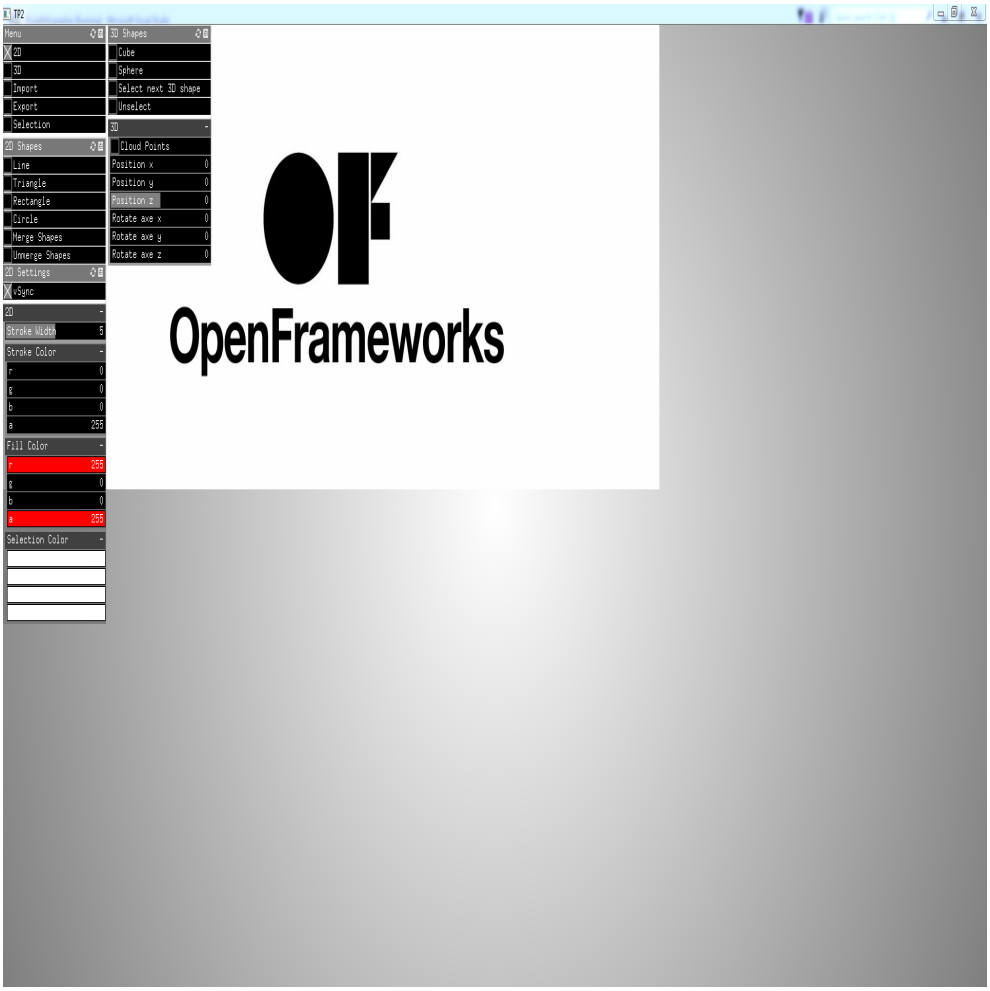


FIGURE 5.2 – Une image importée dans le renderer 2D.

### 5.1.2 Exportation

Pour satisfaire ce critère, l'application doit pouvoir exporter des fichiers images (dans notre cas une image .png) et de la sauvegarder sur l'ordinateur utilisé.

L'application permet cette fonctionnalitée grâce à l'utilisation du bouton "Export". Celui-ci ouvre un navigateur de fichier qui permet à l'utilisateur de choisir facilement l'emplacement de sauvegarde ainsi que le nom du fichier sauvegardé. Cette image est celle rendue sur le renderer 2D.

Les figures suivantes demontrent cette fonctionnalitée.

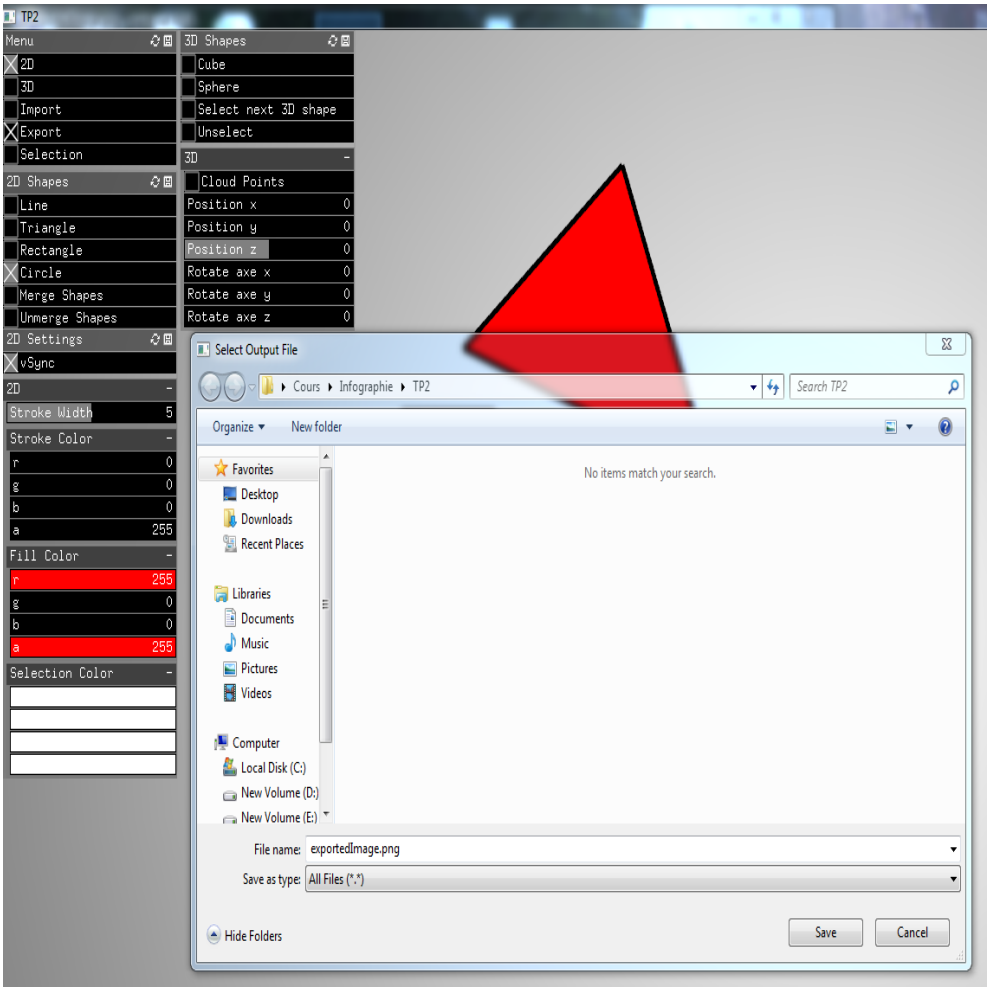


FIGURE 5.3 – Le bouton Export et son navigateur.

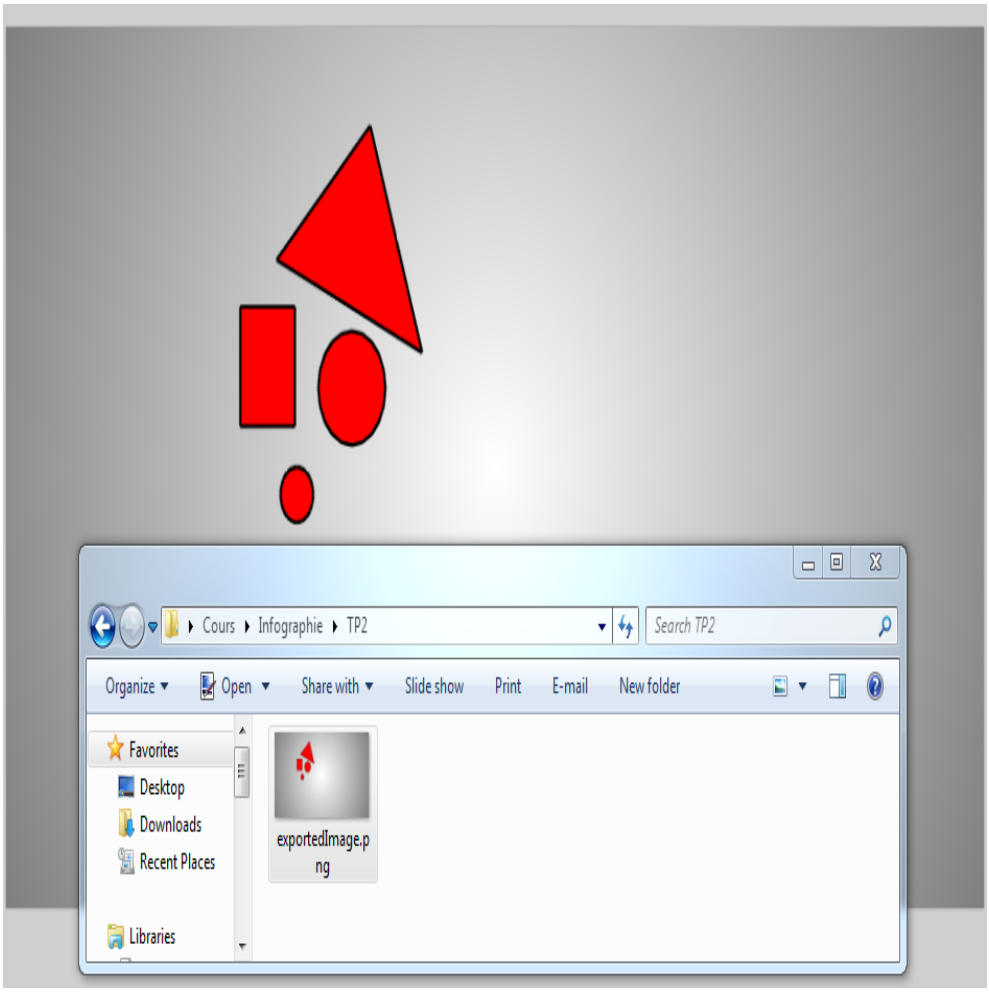


FIGURE 5.4 – Une image exportée à partir de l'application Paint3D+.

## 5.2 Dessin vectoriel

## 5.3 Transformation

### 5.3.1 Structure de la scène

Toutes les entités géométriques présentes dans une scène sont organisées dans une ou des structures de données qui permettent de gérer les transformations et le rendu graphique de chaque élément.

Pour le mode 3D de l'application, il est possible de sélectionner chaque Primitive 3D en appuyant sur le bouton "Select next 3d shape". Une fois sélectionnée, il est possible de sa

couleur de remplissage, sa position en x, y, z, son orientation en x, y, z, de la supprimer et de la transformer en nuage de points.

Cela est possible grâce à une structure vector<Obj3D*> qui contient toutes les classes dérivées de Obj3D comme le cube ou la sphère. Voici un exemple de code le montrant :

```
27  class Renderer2D;
28  class Obj3D;
29  class Renderer3D;
30  class RendererModel;
31
32  class ofApp : public ofBaseApp{
33      public:
34
35          std::vector<Coord> m_buffer;
36          std::vector<Obj2D*> m_obj2DVector;
37          std::vector<Coord3D> m_buffer3D;
38          std::vector<Obj3D*> m_obj3DVector;
39          AppState m_state;
40          AppMode m_mode;
41          int m_clickRadius;
42          bool isTakingScreenshot;
43          bool isClearingButtonsShapes, isClearingButtonsModes;
44          bool m_firstTimeSelection;
45          int m_selectionIndex;
46
47          ofApp();
48          ~ofApp();
49          void exit();
50
```

FIGURE 5.5 – Structure de données stockant les entités 3D.

```
785 void ofApp::buildTriangle() {
786     m_obj2DVector.push_back(new app::Triangle(m_buffer, 0, renderer2d->strokeWidth.get(), renderer2d->colorStroke.get(), rende
787 }
788
789 void ofApp::buildCircle() {
790     double radius = calculateDistance(m_buffer[0], m_buffer[1]);
791     m_obj2DVector.push_back(new app::Circle(m_buffer[0], radius, 0, renderer2d->strokeWidth.get(), renderer2d->colorStroke.get(
792 }
793
794 void ofApp::buildLine() {
795     m_obj2DVector.push_back(new app::Line2D(m_buffer, 0, renderer2d->strokeWidth.get(), renderer2d->colorStroke.get(), rendere
796 }
797
798 void ofApp::buildCube() {
799     m_obj3DVector.push_back(new app::Cube3D(m_buffer3D, renderer2d->strokeWidth.get(), renderer2d->colorStroke.get(), renderer2
800 }
801
802 void ofApp::buildSphere() {
803     m_obj3DVector.push_back(new app::Sphere3D(m_buffer3D, renderer2d->strokeWidth.get(), renderer2d->colorStroke.get(), rendere
804 }
805
806 double ofApp::calculateDistance(Coord p_coord1, Coord p_coord2) {
807     double x2 = pow((p_coord1.getX() - p_coord2.getX()), 2);
```

FIGURE 5.6 – Méthode montrant comment on stock une nouvelle entité 3D dans vector<Obj3D*>.

### 5.3.2 Transformation interractive

Il est possible de modifier interractivement la translation, la rotation et le redimensionnement de tous les objets présents dans la scène.

En mode 3D, il est possible de changer la position x, y, z, l'orientation en x, y, z et le redimensionnement grâce aux paramètres glissant de la fenêtres "3D Settings".

Voici le images le montrant :

## 5.4 Géométrie

### 5.4.1 Particules

Pour respecter ce critère fonctionnel, l'application doit rendre un nuage de points en une seule commande d'affichage.

Dans l'application Paint3D+, il y a un bouton nommé "Cloud Points" qui permet de transformer un objet 3D sélectionné en un nuage de points où chaque point est un sommet d'unedes faces qui compose l'objet en 3D.

Donc, pour pouvoir utiliser cette option, il faut créer un objet 3D avec le bouton "Cube" ou "Sphere" en mode 3D, sélectionner un des objets 3D créé avec le bouton "Select next 3D Shape" et appuyer sur le bouton "Cloud Points" pour rendre l'objet 3D sélectionné en tant que nuage de points.

Les 2 figures suivantes montrent une sphère rendue sans nuage de points et une sphère rendue avec un nuage de points.

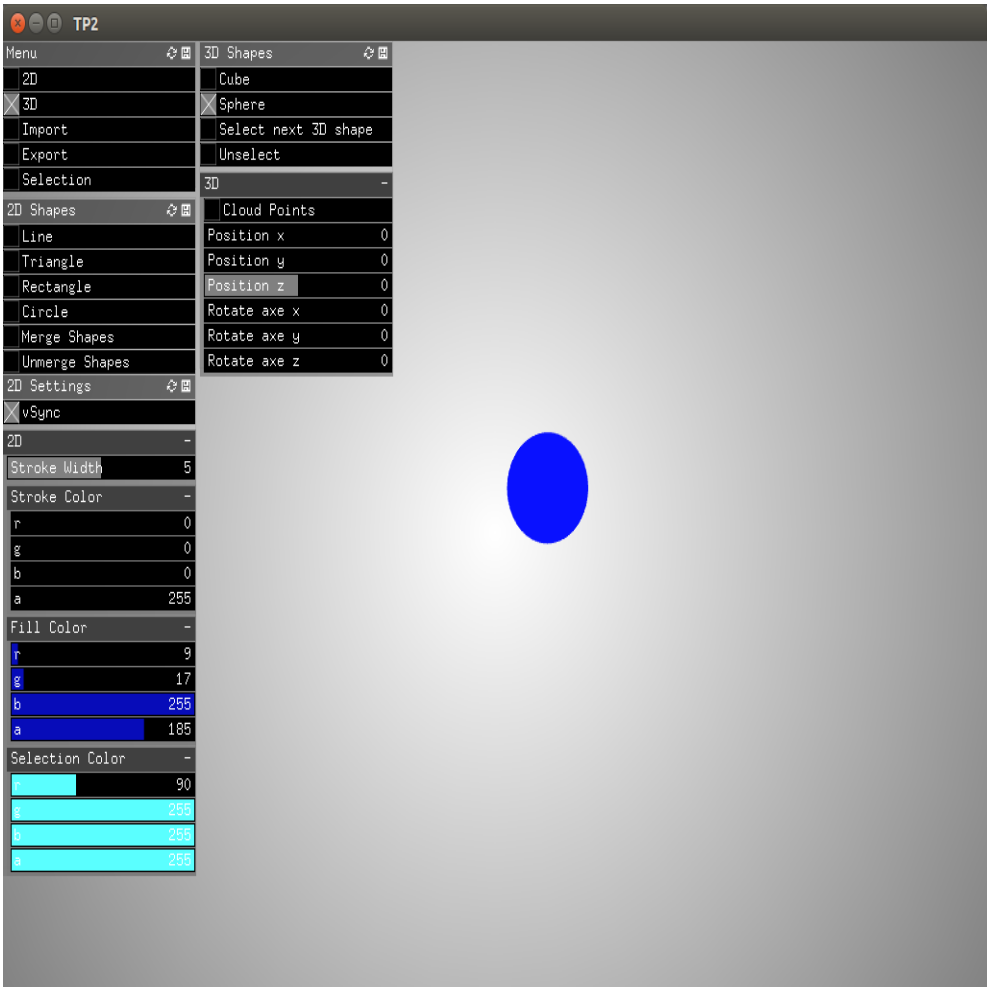


FIGURE 5.7 – Une sphère rendue sans nuage de points.

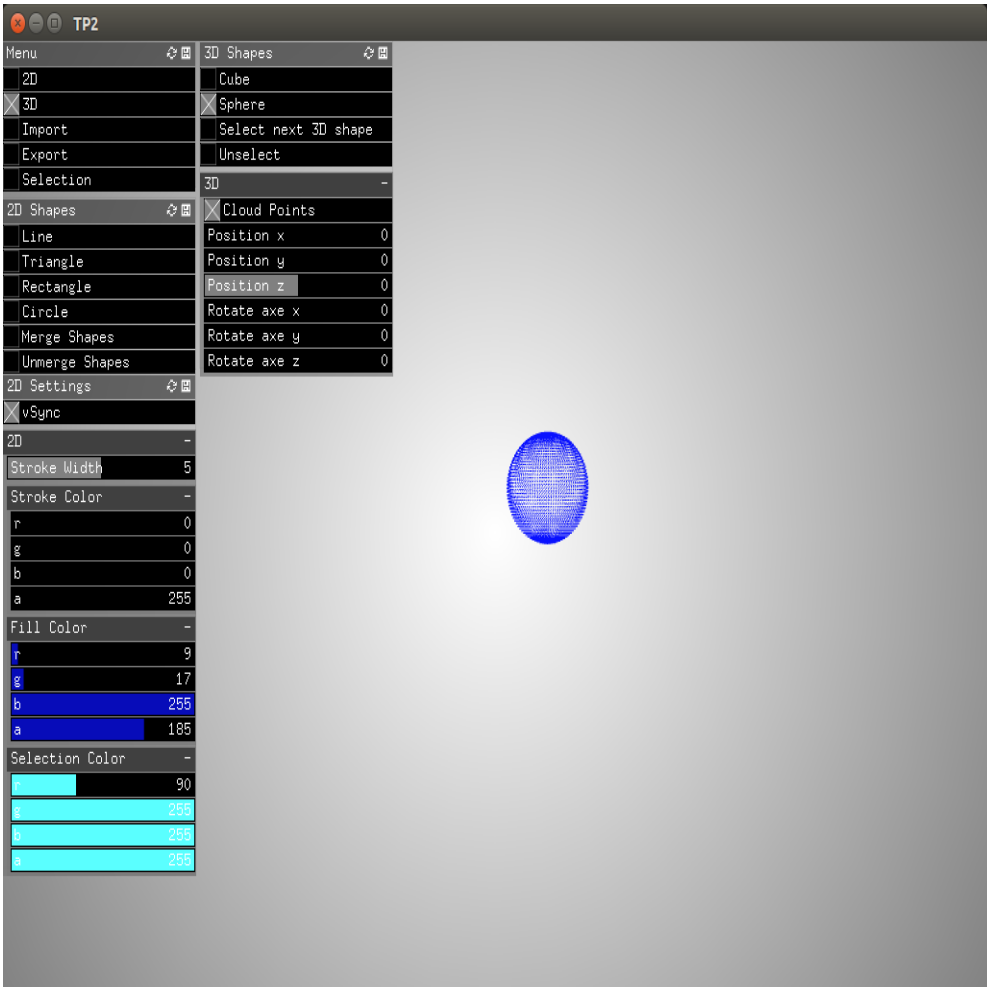


FIGURE 5.8 – Une sphère rendue avec un nuage de points.

### 5.4.2 Primitives 3D

L'application doit permettre de créer au moins 2 types de primitives géométriques 3D à partir d'un algorithme sans aucune données externes à l'application.

L'application Paint3D+ permet de créer 2 primitives 3D localement sans données externes, soit le cube et la sphère. Ces 2 primitives 3D sont créées par composition à partir de 2 classes d'OpenFrameworks, soit OfSpherePrimitive et OfxBoxPrimitive.

Pour créer ces 2 primitives 3D, il faut d'abord sélectionner le mode 3D dans le menu. Ensuite, il faut cocher soit Cube ou Sphere, selon la primitive 3D à créer et cliquer gauche sur la souris à l'endroit où l'on veut que la primitive 3D apparaisse. Une fois crée, chaque primitive 3D a une profondeur par défaut de 30 unités. Cela permet de mettre en valeur l'ef-

fet de perspective et bien montrer qu'il s'agit d'un objet en 3D. Ensuite, pour modifier une primitive 3D créée, il faut la sélectionner avec l'option "Select next 3D shape" et modifier les paramètres modifiables, soit la position en x,y,z l'orientation en x,y,z ou la couleur de sélection ou de remplissage. De plus, lorsqu'une primitive 3D est sélectionnée, il est possible de l'effacer en appuyant sur la touche "delete".

Il est à noter que la rotation est faite à partir de quaternions dans le code.

Les 2 figures suivantes montrent les 2 primitives 3D que l'application peut créer sans données externes.

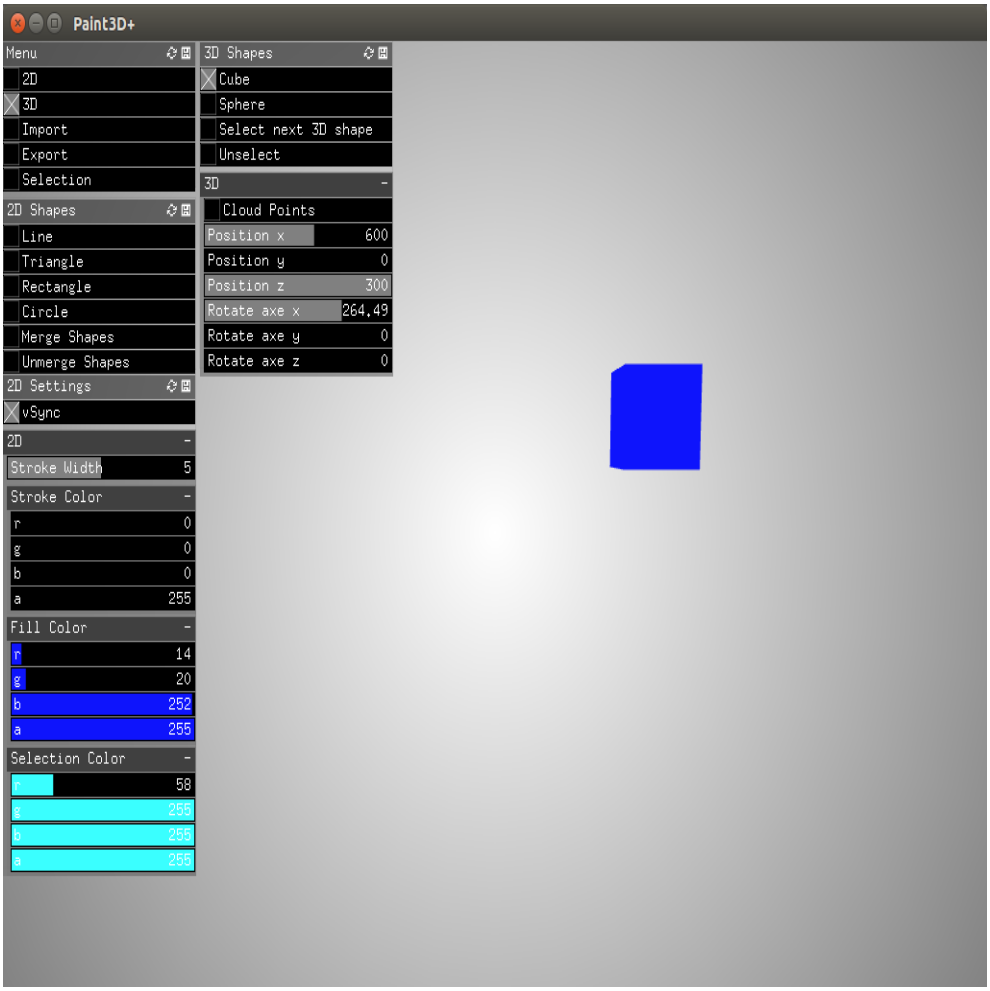


FIGURE 5.9 – Une primitive 3D d'un cube.

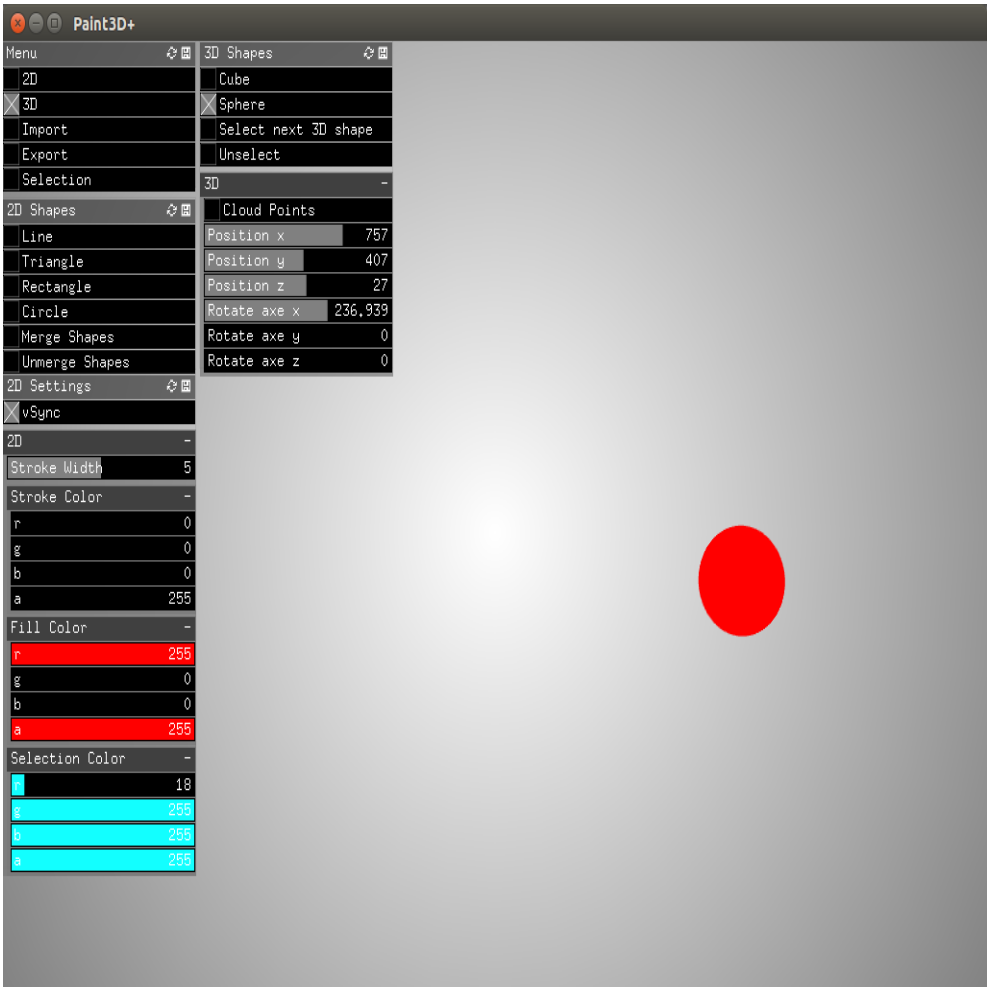


FIGURE 5.10 – Une primitive 3D d'une sphère.

### 5.4.3 Modèle

L'application permet de créer une ou des instances de modèles 3D à partir d'un maillage géométrique importé à partir d'un fichier externe.

Dans l'application Paint3D+, bla bla bla

## 5.5 Illumination

### 5.5.1 Lumière ponctuelle

Possibilité d'avoir au moins une lumière ponctuelle dont la couleur et la position sont prises en compte lors des calculs d'illumination des surfaces présentes dans une scène.

Dans l'application Paint 3D+, bla

### 5.5.2 Matériau

Toutes les entités visuelles présentes dans la scène ont au moins un matériau par défaut qui permet de bien les distinguer si elles sont visibles du point de vue de la caméra.

Dans l'application Paint 3D+, bla

### 5.5.3 Modèle d'illumination

Le rendu des entités visuelles est fait à partir d'un modèle d'illumination qui supporte au moins une combinaison de lumière dynamique et de matériau.

Dans l'application Paint 3D+, bla

# Ressources

Test

# Présentation

Test