Sujet Projet Informatique:
Circuit électrique 3D

2011

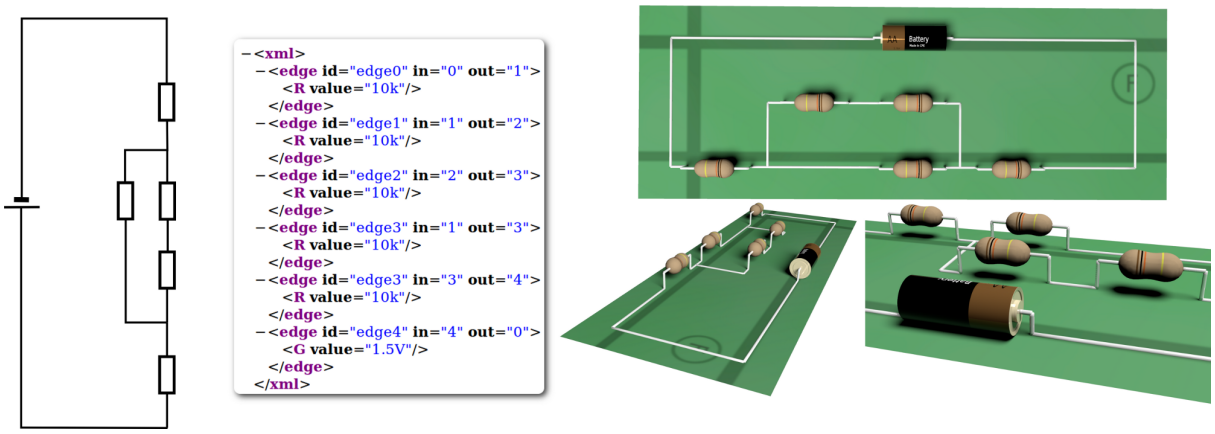

FIGURE 1 – Principe du projet. Gauche : Schéma à modéliser. Milieu : Description formelle analysable. Droite : Visualisation 3D correspondante au schéma électrique.

1 But

L'objectif de ce TP est de réaliser un analyseur et un visualiseur 3D de circuit électrique. Le sujet se décompose en deux parties :

1. Analyse et etude (calcul des tensions et courants) d'un schéma électrique.
2. Visualisation 3D du circuit électrique.

L'étude et la résolution numérique des valeurs de tensions et courants sur un schéma numérique sont la base des logiciels de modélisation électroniques (Pspice, ...). La représentation visuelle aide à la compréhension du circuit, ainsi qu'à postériori pour des simulations plus précise nécessitant l'information géométrique (diffusion de la chaleur, ...).

2 Fonctionnement du projet

Un schéma électrique est donné en tant qu'entrée du projet. On pourra supposer que celui-ci ne contient que des éléments simples : générateurs, résistances.

2.1 Analyse du schéma électrique

Un schéma électrique peut se voir comme un graphe cyclique orienté dont les noeuds contiennent les caractéristiques des composants. La description textuelle de sauvegarde des schémas sera réalisée en format *xml*.

Dans la suite, on considère l'exemple du schéma montré en fig. 1. On pourra utiliser le langage de description [1] suivant en correspondance avec l'indexation proposée en fig. 2.

```
<xml>
 <edge id="edge0" in="0" out="1">
  <R value="10k"/>
 </edge>

 <edge id="edge1" in="1" out="2">
  <R value="10k"/>
 </edge>

 <edge id="edge2" in="2" out="3">
  <R value="10k"/>
 </edge>

 <edge id="edge3" in="1" out="3">
  <R value="10k"/>
 </edge>

 <edge id="edge3" in="3" out="4">
  <R value="10k"/>
 </edge>

 <edge id="edge4" in="4" out="0">
  <G value="1.5V"/>
 </edge>
</xml>
```

La signification des mots clés est la suivante :

- *edge* : désigne le démarrage de la description d'une structure reliant deux noeuds du schéma. Chaque arête possède un noeud d'entrée et un noeud de sortie.
- *id* : identifiant unique d'une arête du schéma.
- *in* : identifiant unique du noeud d'entré de l'arête courante.
- *out* : identifiant unique du noeud de sortie de l'arête courante.
- *R* : Noeud contenant une résistance.
- *G* : Noeud contenant un générateur de tension.

Ce graphe peut se représenter sous forme matricielle M (matrice d'incidence [2]) vu comme l'application de la loi des noeuds. Pour chaque sommet, on désigne l'indice de l'arête incidente et sortante pondéré par la valeur signée de la résistance lui correspondant.

1. Un graphe est classiquement défini localement par ses arêtes, ou connectivités
2. duale de cette description par arête

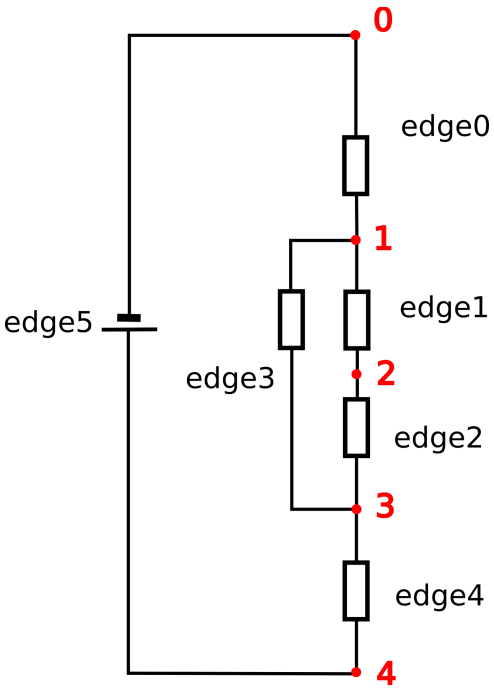

FIGURE 2 – Modélisation du schéma électrique. Les noeuds du graphe de description correspondent à des positions du schéma (points rouges), alors que les arêtes correspondent à un composant du montage.

Soit U le vecteur des tensions inconnues aux arêtes. On notera donc $V = (v_0, v_1, v_2, v_3, v_4)$ dans ce cas particulier. Notons $U = (0, 0, 0, 0, E)$, avec E la tension du générateur situé sur l'arête 5. L'application de la loie des noeuds (matrice d'incidence pondérée) et de la loi des mailles (recherche de cycle dans un graphe) permet d'écrire le système

$$\mathrm{M}V = U \ ,$$

$$\mathrm{M} = \begin{pmatrix} -1/R & 1/R & 0 & 1/R & 0 \\ 0 & -1/R & 1/R & 0 & 0 \\ 0 & 0 & -1/R & -1/R & 1/R \\ 0 & 1 & 1 & -1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix} \tag{1}$$

Une fois le système matriciel rempli automatiquement, l'utilisation d'un algorithme de résolution de système linéaire numérique permet d'obtenir en chaque arête la valeur de la tension associée, et donc du courant.

2.2 Visualisation du schéma

La visualisation du schéma en 3D se réalise à l'aide de la librairie graphique OpenGL. La manipulation à la souris de la visualisation pourra se faire avec l'aide de la librairie *glut*, ou directement sout *Qt*. Un programme minimaliste de manipulation de scène 3D vous est donné en tant que point de départ.

Dans la boucle d'affichage principale réactualisée à vitesse interactive, l'appel explicite à l'affichage d'un triangle se réalise suivant la syntaxe :

```
double x1,y1,... , z3; <- valeurs des coordonnées du triangle

glBegin(GL_TRIANGLES);//debut affichage triangle
glVertex3d(x1,y1,z1); //sommet 1
glVertex3d(x2,y2,z2); //sommet 2
glVertex3d(x3,y3,z3); //sommet 3
glEnd();
```

De manière similaire, on peut afficher un quadrangle directement :

```
glBegin(GL_QUADS);//debut affichage triangle
glVertex3d(x1,y1,z1); //sommet 1
glVertex3d(x2,y2,z2); //sommet 2
glVertex3d(x3,y3,z3); //sommet 3
glVertex3d(x4,y4,z4); //sommet 4
glEnd();
```

Le cas d'un triangle dont la normale est constante sur la portion de plan est obtenu par l'appel :

```
glBegin(GL_TRIANGLES);//debut affichage triangle
glNormal3d(nx,ny,nz); //normale du triangle
glVertex3d(x1,y1,z1); //sommet 1
glVertex3d(x2,y2,z2); //sommet 2
glVertex3d(x3,y3,z3); //sommet 3
glEnd();
```

Lorsque plusieurs triangles sont affichés cotes à cotes (maillage), on cherche à donner l'apparence d'une surface globalement lisse. Les normales ne sont alors plus considérée comme constante sur un patch linéaire élémentaire (triangle ou quad), mais comme une donnée variable associée à chaque sommet. Les sommets sont alors vus comme l'échantillonnage d'une fonction lisse dont l'information de normale est connue en ces points particuliers.

Un triangle dont les sommets sont associés à des normales différentes possède la syntaxe suivante :

```
glBegin(GL_TRIANGLES);//debut affichage triangle
glNormal3d(nx1,ny1,nz1); //normale du triangle 1
glVertex3d(x1,y1,z1); //sommet 1
glNormal3d(nx2,ny2,nz2); //normale du triangle 2
glVertex3d(x2,y2,z2); //sommet 2
glNormal3d(nx3,ny3,nz3); //normale du triangle 3
glVertex3d(x3,y3,z3); //sommet 3
glEnd();
```

Finalement, l'affichage d'un ensemble de triangles dont les normales sont définies par sommet suivra la syntaxe suivante :

```
glBegin(GL_TRIANGLES);
for(int k_triangle=0;k_triangle<N_triangle;++k_triangle)
{
  //appels à glNormal3d et glVertex3d
  // des triangles correspondants
  // ...
}
glEnd();
```

Une résistance pourra ainsi être modélisée par un cylindre formé de triangles ou de quadrangles. On pourra ainsi former une librairie d'éléments de bases pouvant être appelés lorsque nécessaire.

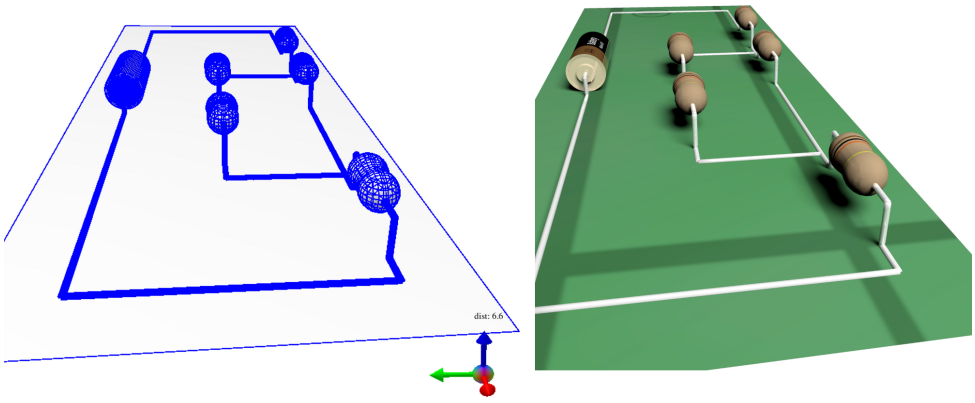

FIGURE 3 – Visualisation possible du schéma électrique. Gauche : Représentation du maillage triangulé sous OpenGL. Droite : rendu final texturé du montage.

3 Travail demandé

Vous réaliserez l'analyse ainsi que la visualisation automatique d'un circuit simple formé de générateur de tension et de résistances. Dans un premier temps, on pourra s'intéresser au cas particulier du schéma fourni en exemple.

Vous implémenterez la lecture d'un fichier de sauvegarde de circuit au format *xml*. Le parsing pourra se réaliser à l'aide de librairies dédiées : *tinyxml* ou *libxml*.

À partir de la lecture de ce type de fichier, vous implémenterez la génération automatique de la matrice modélisant la loi des noeuds de votre schéma. La résolution numérique de cette matrice pourra se faire à l'aide de bibliothèques dédiées : *eigen*, *newmat*, ...

Enfin, vous réaliserez l'affichage 3D de votre circuit à l'aide de modèles simples. Dans un premier temps, vous mettrez en place des modèles géométriques simples (cylindres, plan, ...) puis tenterez d'itérer sur des modèles de visualisation plus détaillés. Les modèles géométriques pourront être pré-stockés une fois pour toutes, et finalement modélisés à l'aide de logiciels spécifiques au besoin (Blender, ...).

Le coeur de votre projet consiste à lire, analyser et visualiser un chemin simple tel qu'illustré dans ce document. Il est indispensable de réussir ces trois étapes dans ce cas particulier. Il est inutile de généraliser ou d'améliorer l'approche si ces trois étapes initiales ne sont pas réalisées.

On notera que le schéma de description ne renseigne pas sur la position 2D des éléments électriques sur la plaque. Vous détaillerez précisément l'approche que vous suivrez. Pistes possibles :

- Supposition d'une structure hiérarchique simple. Le placement peut alors se réaliser en incrémentant des offsets en (x, y) en fonction du parcours du circuit. (Approche la plus simple, mais non applicable pour des graphes complexes).
- Utilisation de logiciels de modélisation de schéma externe de type PSpice et lecture de leur fichier de sauvegarde.
- Utilisation de logiciels de visualisation de graph et analyse de leur format de sortie (*graphviz*).

Le projet fera potentiellement appel à plusieurs librairies externes. Vous expliciterez avec précision l'utilisation à ces librairies et la raison de vos choix. Vous prêterez également un soin particulier à préciser les limitations de votre approche : quels circuits sont modélisables, quels circuits ne le sont pas.

L'ensemble de votre code devra faire apparaitre une séparation claire entre données graphiques et caractéristiques internes. On pourra par exemple créer des interfaces permettant d'éviter de mélanger le traitement des données de la partie graphique.

```
//interface d'objets 3D
class drawable3D
{
public:
  virtual void draw() const=0;
};

//Objets positionnable dans un schéma
class embeeded_object
{
public:
  Position2D getPosition() const;
  void setPosition(const Position2D& p);
private:
  Position2D p;
};

//données interne au composant
class Resistor
{
 private:
  int value;
};

class Resistor_scheme: public Resistor, public embeeded_object;
class Resistor3D : public Resistor_scheme, public drawable3D
{
 public:
   virtual void draw() const;
};
```

4 Amélioration possible

4.1 Ajout de composants

Il vous est possible de réfléchir à l'ajout de composants linéaires à vos schémas. La résolution numérique des tensions et courants dans le cas statique n'en est pas affectée, mais la visualisation peut prendre en compte ces éléments supplémentaires.

Il est également possible de modéliser des éléments non linéaires (ex. diodes, transistors, ...). La résolution d'un système linéaire unique n'est plus suffisante. Il devient alors nécessaire de réaliser plusieurs résolutions (dans le cas du modèle linéaire par morceaux de la diode), ou de modéliser celui-ci par une fonction non linéaire. La recherche de la solution se réalise alors à l'aide de méthodes numériques (dichotomie, méthode de Newton, ...).

4.2 Évolution dynamique ou régime permanent

On peut dans un premier temps modéliser le système en régime oscillatoire périodique permanent. Dans ce cas, on considèrera une matrice complexe pondérée par les impédances des éléments.

Dans le cas d'un régime forcé non sinusoïdal périodique permanent, on pourra approximer la solution suivant une décomposition de l'entrée sous forme de série de Fourier.

Enfin, dans le cas dynamique non permanent, il devient nécessaire de modéliser les éléments par leur équation différentielle. La modélisation de cette équation pourra se faire par assemblage des équations différentielles associées à chaque élément. Enfin, la résolution de l'évolution temporelle des tensions et courant se réalisera par incrément numérique suivant un pas de temps Δt. C'est ce type d'approche qui est réalisé dans les logiciels de simulations complets.

4.3 Modélisation de schémas

On pourra réfléchir à la mise en place d'une interface de modélisation permettant de réaliser des schémas manuellement. On pourra s'inspirer d'outils tels que *PSpice*, ou de dessins vectoriels tels que *XFig*. Enfin, vos schémas pourront être sauvegardés sous format échangeable et lisible par votre simulation et visualiseur 3D.