Pseudocode non récursif

```
// A et B sont les 2 textures d'entrée

If ( !shedderi.useNeighbors && !shedderi'.useNeighbors)
   {
       transition( programe( programe(A,shedderi) ,shedderj) , programe(
programe(A,shedderi'),shedderj')
   }
Else
   {
       // Pour l'entrée 0
       If (!shedderi.useNeighbors)
       {
           A0i = programe(A,shedderi)
           A0ij= programe(A0i,shedderj)
       }
       Else
       {
            A0ij= programe( programe(A,shedderi) ,shedderj)
       }


       // Pour l'entrée 1
       If (!shedderi'.useNeighbors)
       {
           A1i' = programe(A,shedderi')
           A1ij'= programe(A1i,shedderj')
       }
       Else
       {
            A1ij= programe( programe(A,shedderi) ,shedderj)
       }

       transition( A0ij, A1i'j')
   }
```

Pseudocode récursif

// A0 et A1 sont les 2 textures d’entrée, L et L' sont respectivement les listes de shaders de l'entrée 0 et ceux de l'entrée 1.

//pour l’entrée 0:

```
Def f(A,L)
{
	Int c=0;
	Texture A1=A;
	Texture A2;
	T=[A1,A2];
	If (L.size()==0)
	{
		return T[c];
	}
	Else
	{
		while (!L[0].useNeighbors && L.size()>0)
		{
			T[c] =  programme(T[1-c], L[0]);
			L=L[1:];
		}
		if (L.size()>0)
		{
			T[c]= programme(T[1-c], L[0]);
			L=L[1:];
		}
		return f(A2,L)
	}
}
```

```
Int n=L.length();
Int i=0
While i<n:
{
if (!L[i].useNeighbors)
       {
       A=programme(A, L[i]
       {
else
       {
```

```
Def f(A,L):
       If size(L)==0:
              return A
       Else:
              While size(L)>1 && !getNeighbors(L[0]):
                     //import dynamique dans L[1] écrire directement dans le programme
                     effet=L[0]
                     L=L[1:]
              f(program(A,effet),L)
```

Où :

- L est la liste des effets
- getNeighbours est une fonction qui à partir d'un effet renvoie true si cet effet utilise des voisins et false sinon
- l'import dynamique remplace l'effet L[1] par une copie de L[1] où on a intégré les qq lignes de codes correspondantes à l'effet L[0]
- program(A,e) applique à A l'effet e et revoit la texture correspondante (enregistrée dans TX1 ou TX2) → il faut sauvegarder le buffer utilisé en dernier pour pouvoir sauvegarder dans l'autre

```
counter++;

import(L[0])
        .then(counter--; obj => { print('Module loaded: ' + obj.code + '
nb uniforms ' + obj.uniforms.length); })
        .catch(counter--; err => { print('Failed to load object: ' +
err); });
```