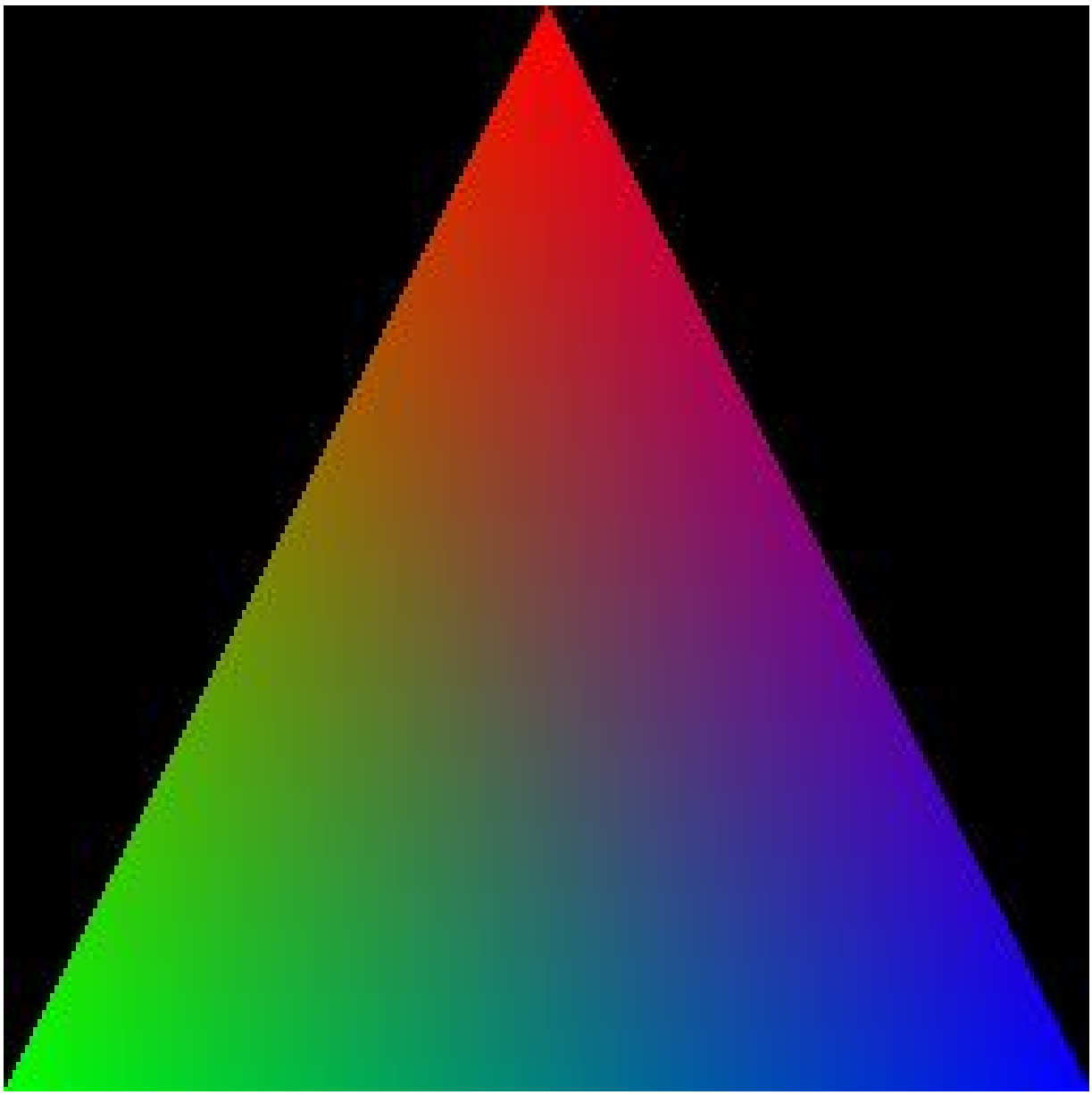
Cpaintpaint



Projet du cours de compilation

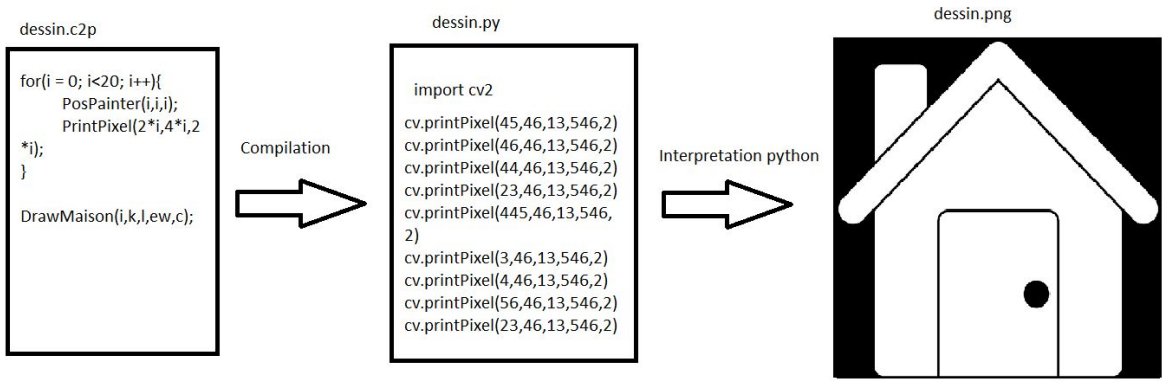
Elèves: Ruedin Cyril
Magnin Lancelot

Date: 05.02.2017

Introduction

Le but de ce projet est de créer un langage qui permet de dessiner avec du code. Ce code est un code similaire en syntaxe au C, comprenant des instructions relativement basique, permettant de dessiner des formes simples.

Le compilateur transforme le code CPaintPaint en directives Python/OpenCV. Le code .py généré est ensuite interprétable en python et génère l'image correspondante. Il crée une image png.



Lancelot Magnin
Cyril Ruedin

Langage

Dépendance

Langage source : CPaintPaint
Langage intermédiaire : Python 3.3
Langage de destination : Liste des pixels en directives OpenCV python
Destination : fichier PNG

Opérateur arithmétique

- Addition (+)
- Soustraction (-)
- Divistion (/)
- Modulo (%)

Opérateur de comparaison

- Egalité (==)
- Inégalité (!=)
- Plus grand que (>)
- Plus petit que (<)
- Plus petit ou égal (<=)
- Plus grand ou égal (>=)

Affectation

Operateur (=)
var = value
Le type des variables est décimal.

Conditions

Permet de comparer deux valeurs avec les opérateurs de comparaison

Exemple

```
if(a<t){ //program }
if((t*2)<100){ program }
```

Boucles

Les boucles permettent exécuter plusieurs fois une suite d'instruction

While

Effectue le programme tant que la condition est vraie

Syntaxe :
while(condition) { programme }

Exemple

Effectue le programme tant que b est inférieur à 50

```
While(b<50){ //program }
```

For

Effectue le programme un certain nombre de fois

Syntaxe :
for(initialisation, comparaison, incrément) { programme }

Exemple

Effectue 100x le programme

```
for(i=0, i<100, i=i+1){ //program }
```

Séparation des instructions

Les instructions dans un même blocs doivent être terminée par un point virgule (“;”).
La dernière instruction d’un bloc ne doit pas être terminée par un point virgule

```
t=40;
for(i=0, i<400, i=(i+t)){
    for(j=0, j<400, j=(j+t)){
      if((i%(t*2))<t){
        if((j%(t*2))<t){
          drawRectangle(i,j,i+t,j+t,255,255,0);
          drawCircle(i+20,j+20,t/2,20,255,240)
        }
      };
      if((i%(t*2))>=t){
        if((j%(t*2))>=t){
          drawRectangle(i,j,i+t,j+t,255,255,255);
          drawCircle(i+20,j+20,t/2,20,255,60)
        }
      }
    }
}
```

En bleu, les dernières instructions et en verts, les instructions non terminale, suivies d’un point virgule.

Séparer les instructions comme en c++ ne fonctionne pas. Le fichier “codeErrors.cptpt” en annexe est un exemple de code non fonctionnel.

Fonctions

Les fonctions permettent de dessiner, les paramètres typés décimal sont automatiquement converties en valeurs entières.
La taille de l'image varie selon les pixels écrits.

Les valeurs négatives placent les pixels en partant de l'autre côté du canevas.
Les valeurs des couleurs sont réduite ou augmentée de 255 afin d'être dans la zone [0;255].

printPixel

Dessine un pixel

Paramètres

Position	Type	Nom	Description
0	Decimal	PosX	Position X du pixel à dessiner (max 512)
1	Decimal	PosY	Position Y du pixel à dessiner (max 512)
2	Decimal	Red	Couleur du pixel, composante RGB rouge
3	Decimal	Green	Couleur du pixel, composante RGB rouge
4	Decimal	Blue	Couleur du pixel, composante RGB rouge

Exemple

Pixel blanc à la position (24;300)

```
printPixel(24,300,255,255,255)
```

drawRectangle

Dessine un rectangle

Paramètres

Position	Type	Nom	Description
0	Decimal	PosXa	Position X du premier coin
1	Decimal	PosYa	Position Y du premier coint
2	Decimal	PosXb	Position X du deuxième coin
3	Decimal	PosYb	Position Y du deuxième coin
4	Decimal	Red	Couleur du rectangle, composante RGB rouge
5	Decimal	Green	Couleur du rectangle, composante RGB verte
6	Decimal	Blue	Couleur du rectangle, composante RGB bleue

Exemple

Rectangle bleu

```
drawRectangle(20,20,90,60,0,0,255)
```

drawCircle

Dessine un cercle

Paramètres

Position	Type	Nom	Description
0	Decimal	PosX	Position X du centre
1	Decimal	PosY	Position Y du centre
2	Decimal	Radius	Rayon
4	Decimal	Red	Couleur du cercle, composante RGB rouge
5	Decimal	Green	Couleur du cercle, composante RGB verte
6	Decimal	Blue	Couleur du cercle, composante RGB bleue

Exemple

Cercle rouge d’un rayon de 60.

```
drawCircle(90,90,60,255,0,0)
```

drawLine

Dessine une ligne

Paramètres

Position	Type	Nom	Description
0	Decimal	PosXa	Position X du point de départ
1	Decimal	PosYa	Position Y du point de départ
2	Decimal	PosXb	Position X du point de d'arrivée
3	Decimal	PosYb	Position Y du point de d'arrivée
4	Decimal	Red	Couleur de la ligne, composante RGB rouge
5	Decimal	Green	Couleur de la ligne, composante RGB verte
6	Decimal	Blue	Couleur de la ligne, composante RGB bleue
7	Decimal	Thinkness	Epaisseur de la ligne

Exemple

Ligne rouge d'une épaisseur de 6

```
drawLine(20,30,60,60,0,255,0,6)
```

Guide utilisateur

Voici un exemple de code ainsi que des instruction pour l'exécuter.

```
for(i=0,i<499,i=i+1){
   j=0;
   while(j<499){
       if((i%20) <10){
           if(j<250){
                printPixel(i,j,150,250,200)
           }
       };
       if(j>250){
           printPixel(i,j,i%250,(10+j)%255,(i+j)%250)
       };
       j=j+1
   }
};
drawCircle(250,250,20,0,50,250);
drawRectangle(300,150,350,350,250,100,100);
drawLine(10,10,300,100,50,250,100,5)
```

Compilation

Pour compiler correctement, les fichier suivants sont nécessaire :

- AST.py
- compiler.py
- lexPaint.py
- parserPaint.py

Un ligne de commande suffit pour la compilation

```
$ python compiler.py programs/superprogram.ptpt
```

Affichage du résultat

Le résultat est un fichier python contenant des directives OpenCV contenant les positions des Pixel.
Pour installer OpenCV

```
$ pip install opencv-python
```

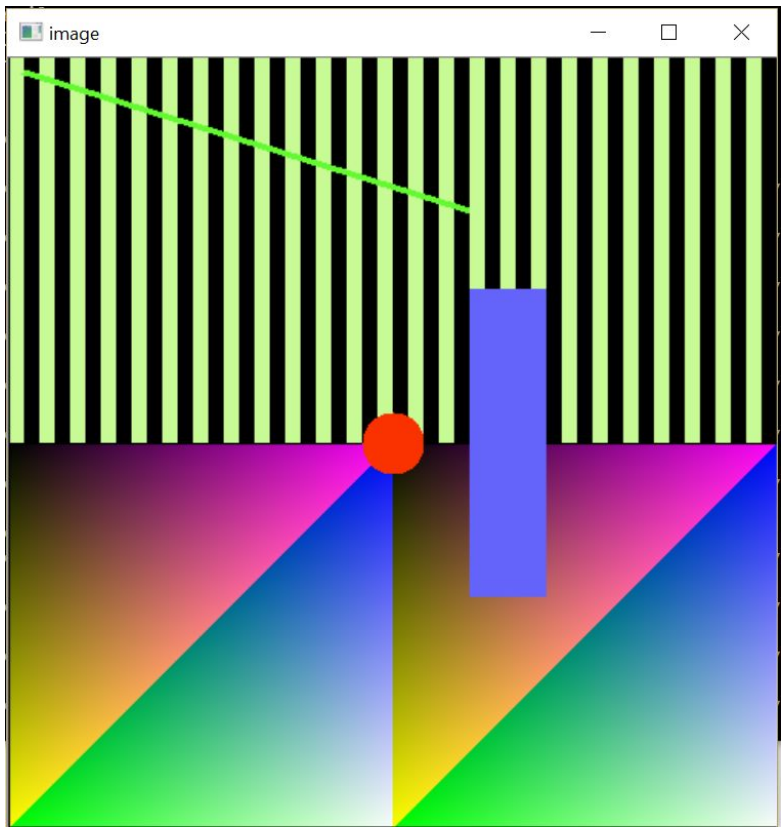
Ou avec le fichier whl (fichier en annex)

```
$ pip install opencv_python-3.1.0-cp35-cp35m-win_amd64.whl
```

Puis pour lancer le programme

```
$ python superprogram.py
```

Le résultat correspond à l'image ci-dessous.



Conclusion

Les objectifs ont été atteints. Le langage permet de dessiner des pixels et de réaliser des dégradés et formes grâce aux boucles et aux opérateurs implémentés. Le langage permet aussi de dessiner des formes plus facilement grâce aux fonctions de plus haut niveau.

Améliorations

Bien que le projet permet de réaliser des dessins relativement complexe, certaines amélioration peuvent être effectuées pour rendre le langage un peu plus viable.

Performances

Les pixels sont à chaque fois réécrit un par un. La performance dépend du nombre de pixel écrit, les boucles et les formes peuvent changer beaucoup de pixels. En cas de réécriture d'un pixel, il n'y pas de mécanisme d'optimisation supprimant l'ancien pixel, ce qui peut créer de lourd fichier, long à exécuter.

Il faudra donc un système d'optimisation qui ne garde que la dernière valeur de pixel. Le format de sortie est lourd aussi, une matrice de couleur aurait été plus performante.

Fonctions mathématique

Les fonctions mathématique plus avancée permettent de dessiner des objets plus complexe très simplement. Par exemple la racine carrée et les puissances permettent de dessiner des cercles simplement. Des outils de dessin comme les courbes de Béziers seraient un must-have.

If..else

Très pratique, par exemple pour faire des changement de couleur

3D

Il serait possible d'ajouter d'autres fonctions de dessins prenant un paramètre z. En fonction de la profondeur (z) on pourrait imaginer rendre les couleurs plus sombres pour donner l'effet de profondeur.
En plus de l'effet de perspective, il faudrait améliorer la performance pour éviter d'avoir des pixels superposés car passer de surface à volume, le nombre de pixel augmente exponentiellement.

Script de compilation + exécution

Un script qui compile et exécute le code et qui ressort directement l'image en png pourrait être plus "user-friendly"

Annexes

Voici les documents qui se trouvent en annexes de ce rapport.
Le github de ce projet se trouve ici: https://github.com/llmmg/Cpaintpaint

Exemple de code OK

- ballsIregular.cptcpt
- chess.cptpt
- degrade.cptpt
- pixeloverride.cptpt

Exemple de code KO

- codeErrors.cptpt