UNIVERSITÉ DU QUÉBEC À MONTRÉAL

Travail Pratique

Documentation du projet
PRÉSENTÉ À L'UNIVERSITÉ DU QUÉBEC À MONTRÉAL
COMME EXIGENCE DU COURS INF600E
À M. ÉTIENNE M. GAGNON

PAR
MATHIEU DES LAURIERS DESM07099809
ZACHARIE CHENAIL-LARCHER CHEZ24069706

DÉPARTEMENT D'INFORMATIQUE
LE 1ER MAI 2022

Présentation du projet

Ce document contient la documentation ainsi que des informations pertinentes concernant le langage *Dessinart*. L'objectif du travail est de construire un langage informatique permettant la création de dessins dans un environnement deux dimensions. Les dessins seront générés à partir d'instructions contenues dans un fichier *.dsa* (raccourci pour *Dessinart*). Ce projet sera créé à l'aide de l'outil *SableCC*[i], tel qu'utilisé tout au long de la session. Le langage utilisera la bibliothèque Swing de Java, qui contient toutes les classes et fonctions nécessaires à la création pour « peindre » des graphiques et des images.

Fonctionnalités implémentées

- Crayon et son déplacement
- Canvas de dessin
- Fonctions et procédures (sketchs)
- Fonctions mathématiques (Sinus, Cosinus, Tangente,…)
- Calculs arithmétiques de base (add, sub, mul, div)
- Opérations conditionnelles et boucles itératives (while, for, etc.)
- Variables (locales et globales), Commentaires
- Types de base (Int, Bool, Color)

Division du travail

Pour la conception de ce langage, nous avons utilisé la fonctionnalité *Code With Me* de l'environnement de développement intégré *IntelliJ IDEA*. Cette fonctionnalité nous permet de programmer tout les deux dans le même environnement. Il est difficile pour nous de séparer qui a fait quoi exactement car nous étions toujours les deux ensembles lors de la programmation. L'utilisation de cet outil à par ailleurs accélérer le développement car il évite ainsi qu'une personne attende la partie de l'autre.

Types de variables

Notre langage comporte trois types de variables différentes. Il contient les variables de type Int (Entier), les variables de type boolean (booléen) ainsi que les variables de type Color.

```
1.  int x = 10;
2.  boolean estVrai = true;
3.  color couleur = ((10,25,30));
```

Description des fonctionnalités

Afin que le programme soit fonctionnel, l'utilisateur doit déclarer un canvas et un cayon au début du programme. Le canvas prend comme arguments la hauteur et la largeur. Le crayon, quant à lui, prend en arguments la taille du trait, ainsi que sa position (en coordonnées x et y).

```
1.  define canvas(250, 250);
2.  define pencil(10, 100, 100)
```

Code 1 : définition du canvas et crayon

Après avoir déclaré le canvas et le crayon, il faut définir la fonction principale. Cette fonction se nomme exclusivement *dessinart*. Son nom est fortement inspiré du nom de notre projet. Cette fonction ne contient pas de parenthèses car elle ne peut contenir de variables.

```
1.  dessinart {
2.      // Code à exécuter
3.  }
```

Code 2 : définition de la fonction principale

Le langage gère également l'utilisation de fonctions et de procédures. Dans le jargon de notre projet, nous avons décidé de nommer les procédures « sketches », qui est un terme anglais qui veut dire « dessiner » ou bien « esquisser ». Les fonctions retournent une valeur tandis que les « sketch » non. Les « sketch » seront plutôt utiliser pour faire des dessins.

La signature des fonctions est la suivante : elle est déclarée avec le mot-clé *func* il faut définir un type de retour (soit int, bool ou color), spécifier le nom de la fonction, spécifier les arguments (aucun, un seul ou plusieurs). Ce qui est situé entre les deux accolades représente le code à exécuter. Il ne faut pas oublier de spécifier le return à la fin de la fonction.

```
1.  func int tripler(int nombre){
2.      int r = nombre + nombre + nombre;
3.      return r;
4.  }
```

Code 3 : Définition d'une fonction

Pour les « sketches », la signature va comme suite : utilisation du mot-clé « sketch », spécifier le nom de du sketch ainsi que ses arguments (aucun, un seul ou plusieurs). Ce qui est situé entre les deux accolades représente le code à exécuter. Le « sketch » n'ont pas de valeur de retour.

```
1.  sketch maison(){
2.     // code à exécuter
3.  }
```

Code 4 : Définition d'un sketch

`

Notre langage contient également des fonctions de base de mathématiques ainsi que de dessin. Dans les fonctions mathématiques, on retrouve entre autres : la fonction *power*, trois fonctions pour arrondir un entier (une pour arrondir au 5^e près, une pour arrondir à la 10^e près et une pour arrondir à la 100^e près). On retrouve également la fonction *sin, cos* et *tan.* Les trois autres dernières fonctions sont des fonctions de génération de nombre aléatoire, soit une pour obtenir un nombre aléatoire en 0 et 255 (pratique pour obtenir une valeur de couleur), une pour obtenir un nombre aléatoire entre 0 et 100 et une pour obtenir un nombre aléatoire entre 0 et 10.

```
1.  power(int :base, int :exposant)
2.  round5(int :valeur)
3.  round10(int :valeur)
4.  round100(int :valeur)
5.  log(int :base, int :valeur)
6.  sin(int :angle)
7.  cos(int :angle)
8.  tan(int :angle)
9.  random255()
10. random100()
11. random10()
```

Code 5 : Utilisation des functions mathématiques

Concernant les fonctions de dessin, celles-ci sont essentielles pour créer des magnifiques chef-d ’œuvres. Elles nécessitent toutes des paramètres, soit des entiers. La fonction *replace* permet de déplacer le crayon à une coordonnée donnée. La fonction *move* permet de bouger le crayon à une distance X et Y par rapport à sa position présentement. La fonction *draw_to* permet de dessiner du point courant jusqu’à la coordonnée X et Y donnée tandis que *draw* dessine jusqu’à de dessiner jusqu’à une distance X et Y par rapport à sa position présentement. Les deux dernières fonctions concernent le crayon. Le première est *setColor*, qui permet de modifier la couleur d’un crayon et *setWidth*, qui permet de modifier la taille du crayon.

```
1.  replace(int : pos_x, int : pos_y);
2.  move(int : pos_x, int : pos_y);
3.  draw_to(int : pos_x, int : pos_y);
4.  draw(int : pos_x, int : pos_y);
5.  setColor(color : couleur);
6.  setColor((int : r, int : g, int: b));
7.  setWidth(int: taille)
```

La fonction setColor est la seule qui peut être appelée avec 2 types de paramètres. Elle prend en paramètres soit une variable de type color déjà instanciée, ou bien 3 valeurs entières qui représentent le taux de rouge, de vert et de bleu.