

28 janvier 2018

Auteur : Lovis Thomas,

Professeur : Tièche François

Génération d’animations

Rapport Animaker

Table des matières

[1. Introduction 2](#_Toc504923369)

[1.1. Aperçu de code 2](#_Toc504923370)

[2. Fonctionnalités implémentées 2](#_Toc504923371)

[2.1. Analyse lexicale 3](#_Toc504923372)

[2.2. Analyse syntaxique 3](#_Toc504923373)

[3. Prise en main 3](#_Toc504923374)

[3.1. Déclarations / Assignations 3](#_Toc504923375)

[3.1.1. Variables de nombres 3](#_Toc504923376)

[3.1.2. Variables objets 3](#_Toc504923377)

[3.2. Affichages 3](#_Toc504923378)

[3.2.1. Console 3](#_Toc504923379)

[3.2.2. Fenêtre d’animation 3](#_Toc504923380)

[3.3. Animations 3](#_Toc504923381)

[3.3.1. rotations 3](#_Toc504923382)

[4. Problèmes rencontrés 4](#_Toc504923383)

[5. Conclusion 4](#_Toc504923384)

# Introduction

Dans le cadre du cours « Compilateur », nous réalisons un programme qui simplifie la génération d’animation en utilisant la bibliothèque PLY de python.

Notre compilateur permet de générer des animations de formes basiques 2D, c’est-à-dire un carré/rectangle, un cercle et un triangle. Les animations possibles sont : la translation et la rotation. Le compilateur génère du code python utilisant la bibliothèque « pygame ».

## Aperçu de code

L’idée de base était de partir d’un code tel que celui exposé :

ball0 = Ball 1020, 20, 50;  
rect0 = Rectangle 0, 1050, 100, 100;  
  
t = 100;  
  
show ball0;  
show rect0;  
  
while(t){  
 print t;  
  
 move ball0 -10 0;  
 move rect0 0 -10;  
  
 show ball0;  
 show rect0;  
  
 t = t - 1;  
};

Ce code va animer une balle, de rayon 50, à l’écran qui va venir depuis la droite et arriver en position (20,20). En parallèle, un rectangle (il fait 100 x 50) va arriver depuis le bas en position (0, 50).

Il s’agit seulement d’un petit exemple des possibilités de ce programme.

# Fonctionnalités implémentées

Tout code soumis au programme Animaker va être traité analyser et interprété de la façon suivante :

* Le code est analysé par des expressions régulières qui vont définir la nature de chaque mot ou caractère. Il s’agit de l’analyseur lexical ;
* Les mots de lexiques (qu’on appelle tokens) sont ensuite filtrer par l’analyseur syntaxique qui va vérifier la grammaire du texte et créer un arbre syntaxique abstrait. Cet arbre va servir à héberger la suite logique des instructions du futur programme animé.
* L’interpréteur va alors pouvoir utiliser l’arbre crée pour écrire le programme dans une fichier annexe et transformer chaque instruction en instruction adaptée utilisant la bibliothèque pygame.

## Analyse lexicale

Les principales choses qui ont été ajoutées par rapport au lexer du cours sont des mots réservés. Les instructions

reserved\_words = (  
 **…,**  
 **'move'**,  
 **'rotate'**,  
 **'show'**,  
 **'Ball'**,  
 **'Rectangle'**,  
 **'Triangle'**,  
 **'PI'**,  
 **'tick'**)

tokens = (  
 **…,**   
 **'COLOR'**)

literals += **','**

## Analyse syntaxique

# Prise en main

## Déclarations / Assignations

### Variables de nombre

Les variables de nombre peuvent servir à faire des calculs ou à servir de condition pour la boucle d’affichage :

a = 10;

b = 5.2;

c = -24;

d = a – b;

while(a){

…

};

### Variables objet

Avec les variables d’objets, on peut déclarer nos formes et garder des références. Cela va nous permettre de jouer avec leur position et de les afficher :

ball0 = Ball 1020, 20, 50;  
rect0 = Rectangle 0, 1050, 100, 100;

triangle0 = Triangle 0, 0, 100, 100, 0, 100;

## Affichages

### Console

L’affichage console est surtout utile pour afficher des indications au programmeur et éventuellement débugger.

a = 10;

print a;

### Fenêtre d’animation

Toutes les variables d’objet peuvent être afficher à l’aide de la fonction show. Cela a pour effet de dessiner la forme dans la fenêtre d’affichage 2D pygame. Dans cet exemple, un rond :

ball0 = Ball 1020, 20, 50;  
show ball0;

## Animations

### Mouvement simple

### Translations

### Rotations

ball0 = Ball 1020, 20, 50;  
rect0 = Rectangle 0, 1050, 100, 100;

t = 100;

La variable « t » est une variable qui indique nombre de ticks, donc de frames d’animations, que l’on veut.

show ball0;  
show rect0;

On termine la phase d’initialisation par un premier affichage.

## Boucles

La boucle while va servir à modifier les valeurs de positions et réafficher les formes. C’est pourquoi on va l’utiliser alégrement. Le test de la variable t renvoi la valeur « False » uniquement si t==0.

while(t){  
…  
};

L’exemple démontre l’application de la condition t pour la while avec un compteur mais rien n’empêche d’établir des algorithmes plus compliqués avec une condition autre que t.

On affiche d’abord en console, à l’aide de print, le valeur du tick actuelle

# Problèmes rencontrés

## Erreurs de parsing

## Problèmes d’import

Le mot clé « parser » utilisé comme nom de fichier pour l’analyseur syntaxique engendre des difficultés d’import. On n’en connait pas la source.

# Conclusion