Crayon (.cry)

a raster graphics creation language proposal

Naman Agrawal (na2603)
Vaidehi Dalmia (vd2302)
Ganesh Ravichandran (gr2483)
David Smart (ds3361)

Introduction

Raster graphics are images created from two-dimensional arrays of hex codes that represent the rectangular grid of pixels we see for many computer graphics. Inspired by this paradigm, the Crayon programming language provides a way for the user to create all sorts of pixelated images (in color) that they can imagine - like the beautiful creations often made on Microsoft Paint, but worse.

Description

More technically, the language will be compiled into LLVM and feature a Java-like syntax. Crayon is centered around its Canvas data structure that represents the aforementioned grid of Pixel values of a set size. Users can then create images by altering these Pixels that store their location on the grid and the color of the corresponding cell. Conventional operators and block statements allow for more complex, algorithmically-based creations to aid in the process. Shapes and basic artistic building blocks like circles and lines can then be used from Crayon's standard library in order to build better works more easily. Programs created in Crayon will then result in bitmap files to view the created image, whether it be a smiley face, a tree, or a blocky Mona Lisa. Further programs could then superimpose one Canvas on another to create melded images, import pre-made Canvas-based pictures and alter them further, produce the negative of a Canvas image, and more.

Simple Data Types

Data Type	**Description/Example**
int	standard integer 2
float	standard float 2.4
char	ASCII character "4"
string	standard string "123"

boolean	0 or 1
null	standard null
hex	standard hex "#000000"

Complex Data Types

Data Type	**Description/Example**
Array	int a[45];
Canvas	Finite-sized two-dimensional Array that holds Pixel values (explained below) at each coordinate. Pixel values are modified by the user to create the image. Canvas banana[20,20];
Pixel	The canvas is made up of many Pixels. Each pixel has a coordinate and colour. Pixel(banana,x,y,hex)

Keywords

for	for x in banana {x.setColor("#FFFFFF")} OR for (int i; i < 5; i++) {banana[i].setColor("#FFFFFF")}
if	if( ) { }
else	else( ) { }
else if	else if( ) { }
void	standard void

Operators

+, -, *, /, =, %, ++, --, ==, !=, >, <, >=, <=, &&, ||, !

Comment operator:

:'(This is a single line comment)

:'(This is a
Multiple line comment)

Sample code

```
:'( This is a program to paint the American flag, with no stars)

Canvas flag[1000, 2000];

flag.set(0, 0); :'(set cursor at the top left corner of canvas)
flag.fill(700, 500, blue); :'(paints blue area for stars...values are
numbers of pixels, not coordinates)

flag.set(800, 500);
flag.replaceColor(white, red); :'(replace white with red)

upperBound = 0;
lowerBound = 99;
for i in flag.section(800, 1999, 0, 999){
      if(i.ycoord <= upperBound){
            i.setColor(white);
      }
      else if i.ycoord > upperBound{
            lowerBound += 200;
            upperBound += 200;
            flag.move(0, 100) :'(move cursor      0 in the x direction,
            100 in the y direction)
      }
}

upperBound = 1000;
lowerBound = 1999;
for i in flag.section(0, 799, 1000, 1999){
      if(i.ycoord <= upperBound){
            i.setColor(white);
      }
      else if i.ycoord > upperBound{
            lowerBound += 200;
            upperBound += 200;
            flag.move(0, 100) :'(move cursor      0 in the x direction,
            100 in the y direction)
      }
}
```

Output