SEE++

Featuring the all power building block:

By Adar Tulloch, Vishrut Tiwari, Winston Zhang, and Jack LaVelle
Motivation

We wanted to build a:

1. Simple C/Java like language
2. Give the user the best support to build art
3. Implement a primitive type that everyone can use and build on

```java
void setup() { // Defines the display window parameters.
    size(700, 400);
    smooth();
}

void draw() { // Defines an action taken if a mouse-button is pressed.
    if (mousePressed) {
        fill(0);
    } else {
        fill(255);
    }
    ellipse(mouseX, mouseY, 80, 80);
}
```
Language Features

- C-like syntax and behavior (variable assignments, operations, scope-rules)

- Advanced types like Pixels, Points, Circles and an adjustable Canvas for the user to render their drawings
Sample programs

```c
int main() {  
    Canvas(5.0, 6.0);  
    return 0;  
}

int main() {  
    Pixel p = Pixel(Point(10.0, 10.0));  
    float test = p.ep1.x;  
    printf(test);  
    return 0;  
}

int main() {  
    CanvasCircle can = CanvasCircle(1000.0, 1000.0);  
    Point pt = Point(500.0, 500.0);  
    Circle px = Circle(pt, 100.0);  
    Point ppt = Point(10.0, 10.0);  
    Pixel pix = Pixel(ppt);  
    Canvas can1 = Canvas(1000.0, 1000.0);  
    can1 -> append() pix;  
    can -> append().circle px;  
    drawcircle(can, "result.svg");  
    return 0;  
}
```
How did we achieve this?

```plaintext
let ptstruct_t = L.struct_type context [ [ float_t ; float_t ] ] in
let pstruct_t = L.struct_type context [ [ ptstruct_t ] ] in
let cstruct_t = L.struct_type context [ [ ptstruct_t ; float_t ] ] in

let canvasonode_t = L.named_struct_type context "canvasonode" in
ignore(L.struct_set_body canvasonode_t [ [ L.pointer_type (canvasonode_t) ];
( L.pointer_type pstruct_t ) ] false);

let canvascirclenode_t = L.named_struct_type context "canvascirclenode" in
ignore(L.struct_set_body canvascirclenode_t [ [ L.pointer_type (canvascirclenode_t) ];
( L.pointer_type cstruct_t ) ] false);

let canvas_t = L.struct_type context [ [ float_t ; float_t ] ]
in

let canvascircle_t = L.struct_type context [ [ float_t ; float_t ];
( L.pointer_type canvascirclenode_t ) ]
in
```

```plaintext
struct point Point(double x, double y) {
    struct point pt;
    pt.x = x;
    pt.y = y;
    return pt;
}

struct pixel Pixel(struct point ep1) {
    struct pixel cv;
    cv.ep1 = ep1;
    return cv;
}

struct canvas Canvas(double x, double y) {
    struct canvas c;
    c.x = x;
    c.y = y;
    c.first = 0;
    return c;
}
```
How did we achieve this?

```haskell
SBinop(A.Canvas, _) as can, op, crv) ->
  let (._can_s) = (match (snd can) with
    SID s -> (expr builder locals can, s)
  ) in
  raise(Failure "improper usage of shoehorn - canvas")

and (_px_s) = (match (snd crv) with
  SID s -> (expr builder locals crv, s)
) in
  raise(Failure "improper usage of shoehorn - pixel")

(match op with
A.Shoehorn ->
  (* construct new node, add it to front of list *)
  let newnode = L.build_alloc canvnode_t "newnode" builder in
  let next_node_ptr = L.build_struct gep newnode 0 "new_pixel" builder in
  ignore(L.build_store (L.const_null (L.pointer_type canvnode_t)) next_node_ptr builder);
  let pixel_ptr = L.build_struct gep newnode 1 "pixel" builder in
  let pxlv = lookup px_s locals in
  ignore(L.build_store pxlv pixel_ptr builder);
  let canlv = lookup can_s locals in
  let headptr = L.build_struct gep canlv 2 "head" builder in
  let oldhead = L.build_load headptr "oldptr" builder in
  ignore(L.build_store oldhead next_node_ptr builder);
  ignore(L.build_store newnode headptr builder); canlv
  _ -> raise (Failure ("improper usage of shoehorn: -> append() " ^
    (string_of_sexp can) ^ " and " ^ (string_of_sexp crv))))
```
How did we achieve this?

```plaintext
A. ShoehornCircle ->
(* construct new node, add it to front of list *)
let newnode = L.build_alloca canvascircle node_t "newnode" builder in
let next_node_ptr = L.build_struct gep newnode 0 "new_circle" builder in
ignore(L.build_store (L.const_null (L.pointer_type canvascircle node_t)) next_node_ptr builder)
let circle_ptr = L.build_struct_gep newnode 1 "circle" builder in
let pxlv = lookup cl_s locals in
ignore(L.build_store pxlv circle_ptr builder);
let canlv = lookup can_s locals in
let headptr = L.build_struct_gep canlv 2 "head" builder in
let oldhead = L.build_load headptr "oldptr" builder in
ignore(L.build_store oldhead next_node_ptr builder);
ignore(L.build_store newnode headptr builder); canlv
|_ -> raise (Failure ("improper usage of shoehornCircle with " ^
 (string_of_sexp can) ^ " and " ^ (string_of_sexp crl))))
```
How did we achieve this?

generatedfiles="$\texttt{generatedfiles} \{\texttt{basename}\}.ll \{\texttt{basename}\}.s \{\texttt{basename}\}.exe \{\texttt{basename}\}.out" && Run "$\texttt{SEEPP}" "$1" "$\{\texttt{basename}\}.ll" && Run "$\texttt{LLC}" -relocation-model-pic "$\{\texttt{basename}\}.ll" "$\{\texttt{basename}\}.s" && Run "$\texttt{CC}" -o "$\{\texttt{basename}\}.exe" "$\{\texttt{basename}\}.s" "\texttt{printbig.o}" "\texttt{draw.o}" "\texttt{drawcircle.o}" "\texttt{svg.o}" && Run "/$\{\texttt{basename}\}.exe" "$\{\texttt{basename}\}.out" && Compare $\{\texttt{basename}\}.out $\{\texttt{reffile}\}.out $\{\texttt{basename}\}.diff
The potential

def line(point1, point2, f, canvas, step):
    if point1[0] < point2[0]:
        slope = (point2[1] - point1[1])/(point2[0] - point1[0])
        max = point2[0]
        x = point1[0]
        y = point1[1]
    else:
        slope = (point1[1] - point2[1])/(point1[0] - point2[0])
        max = point1[0]
        x = point2[0]
        y = point2[1]

    if slope == 0:
        b = point2[1]
    else:
        b = point2[1]/(slope-point2[0])

    while x <= max:
        if not (x,y) in pixels:
            pixels.append((x,y))
            px = "Point pt" + str(abs(int(x))) + str(abs(int(y))) + " = Point(" + str(abs(int(x))) + ",0", + str(abs(int(y))) + ",0);"
            pixel = "Pixel px" + str(abs(int(x))) + str(abs(int(y)))+ " = Pixel("+"pt" + str(abs(int(x))) + str(abs(int(y))) + ");";
            shoehorn = canvas + " -> append() px" + str(abs(int(x))) + str(abs(int(y))) + ";"
            f.write(px + pixel + shoehorn + "n")
        x += step
        y = x*slope + b
Scope for further improvement

We wanted to build a:
1. Make one Shoe Horn operation
2. Add built-in shapes like squares, lines, triangles etc.
3. Compose multiple canvases onto each other to create complex images
4. Add color
Demo Time