LGA - Language for Graphic and Animation Proposal

Hang Qian (hq2124)
Yuanli Dong (yd2270)
Pindan Hao (ph2389)
Tian Xia (tx2126)

September 25, 2013

1 Motivation

Web is becoming a huge thing. For users and even sometime developers, it may feel like suddenly, web pages can do so many cool things. Everyone in our team was obsessed with LOGO, a language and platform designed for kids that draws simple shapes by a turtle (Ninja!) using code with simple syntax. Though LOGO's era has long passed, with all HTML5 stuff got more and more attention, creating web pages with graphical objects and animation has never been this interesting.

Indeed, with the support from the awesome WebGL, there are quite a few libraries that make their efforts to make the job easier. But we are thinking of taking a language-level approach.

We propose LGA as our project for this course. LGA stands for Language for easy manipulation of Graphics and Animation. And also, we are from NYC. We see LGA as a good choice for developers who want to create simple graphical objects and animation without the pain of wrestling with all complex WebGL APIs and javascript pitfalls and gotchas. With LGA, the job should be smooth, painless and all compatible with existing web technologies.

2 Key features

- LGA focuses on creating 2D objects and animation.

- User can use built-in shapes or creating their own by combining built-in shapes and other user defined shapes (shapes are nested structured).

- Complex movement sequence is created by combination of basic movements, some common movements (e.g. circling) are also provided by the language.

- Movement is independent from certain types of objects, once one movement is defined, it can be used by different objects.

- Sub-elements of a complex object can have their own movement pattern, but they also make movements as the whole object moves (e.g. Two wheels of a bike rotate, but they also move forward because the bike is moving forward).

3 Language Description

LGA, shares a C like syntax with minor tweaks, has simple syntax which could powerful support manipulation of graphics and animation. We design LGA to make it easy for users to make simple graphics and animation. In terms of functionality, we are seeing LGA go with a WebGL approach (as javascript is our goal target language) while providing an easier syntax and wrapping to programmer.

Based on understanding of elements establishing graphics and animation, we provide some basic shapes, motion types and other elements in LGA with which users could easily create their own complex graphics and animation. Compared with other languages like JavaScript, LGA provides users an easier way to manipulate graphics and animation.

Built-ins:

Basic data types

Object type
Move type
Normal functions(closure)
Basic shapes as line(), arc(), curve(), rec()

Data types

Integer - int
Float number - double
Boolean - bool
Color (optional, as we can use a tuple if we set up with RGB) - color

Composite

Array
Tuple
HashMap

Iteration

For loop
While loop

Conditionals

If
Else
Continue
Break
Switch (optional, we are not sure yet if we want it)

4 Examples

LGA provides a simple but powerful set of basic graphics and action types with which users could create more complex graphical objects with motion. The following is an example how a simple animation of a moving bicycle could be created with LGA.

```
//A object named Bike is built first
//A set of x-axis and y-axis will be privately setup for this object
obj Bike {
       /* A variable named frontwheel is created;
        * Wheel is an object created by user with the basic graphical
        * types provided by LGA;
        * The position of the wheel is determined by the coordinates
        * (1, 1);
        * Position for var in a object is a relative position to the
        * object
        */
       var frontwheel = new Wheel(1, 1);
       var backwheel = new Wheel(3, 1);

       // Similar to wheel, a var named frame is created
       // from a user-created object Frame
       var frame = new Frame(1, 2);

       // function run determines how an object will act in animation
       // we can override with our definition
       var run = func ()
       {
              // vars are divided into different start orders to be
              // managed in groups
              startorder s1 = [frontwheel, backwheel];
              /*
               * vars with the same start order could run together with
               * a simple call;
               * For this example, two Wheels will start rotating
               */
              s1.run();
              /*
               * A motion object called Move will be provided with some
               * built-in motions;
```

```
                * users could create their own motion types from Move
                * too;
                * In this example, shift is a motion created by users
                * conducting a certain shifting with a predetermined
                * velocity
                */
               shift();
       }
}

//Object Bike is instantiated to a variable named myBike
var myBike = new Bike();
myBike.run();

//A timer is allocated to control motion of myBike
timer(5, func () {myBike.stop(); myBike.dismiss()});
```