The EasyPet Programming Language Proposal

Team 13

Bo Liu (bl2400)
Liang Wei (lw2425)
Shen Wang (sw2613)
Weikang Wan (ww2267)
Xiaolong Jiang (xj2127)

1. Language Description

The programming language EasyPet provides a way to easily create a 2-D desktop based electronic pet game. Users of this language can design specify the pet, and status that will appear in this game, and they can design the general plot of the game that tie these together. This programming language will be ideal for creating 2-D desktop electronic pet game.

2. Inspiration of our Language

Electronic pet (ePet) is very popular among young people nowadays, and many young people want to design an ePet by their own preference. However, it may be difficult for one to design such kind of applications in C or Java. In our language, we define specific data type and structures to assist user define function modules, which will make it really easy for one to design a personalized ePet by himself.

2.1 Ease to use

A programming language that can be programmed easily to design ePet and which leverage today's standard practice. To design such a kind of application, programmer use Java or C++. However, we thought it would be difficult for someone who is not skilled at Object-Orient programming languages to develop such a specific application. In order to make the language more comprehensible, we design our language as closely to Java as possible.

EasyPet defined several specific data types and a bunch of function modules that allow user to declare an ePet, as well as the status of the pet, and particular action of pets. developers can define the functions whatever they want. They can choose to design different kinds of functions to implement the logic of the operation of this popular game. For example, the developers can design a little game to train the cats, such as Cat and Mouse, fishing, Climbing trees.

2.2 Generic design for ePet

In our language, we define a new type named Pet.Developer can “Pet” to declare different types of the pet easily,like panda, cat. They can just assign values to “Pet”, then it will generate different types of pets.For example, attributes like name, type, height, weight and etc.can be used by users directly.

2.3 Convenient for UI Generation

Developers can get rid of the design of UI. We all know that to design a GUI with JAVA is very tedious. With the help of our language, developers can implement a GUI easily. All the images, backgrounds, and buttons will all be handled by our language.

To sum up, our language is easy to learn,easy to use and especially convenient for developers to design all kinds of electronic pets.

3. Language Elements

3.1. Declarations

The program is primary structured around declarations. The user will create declarations for Pet, and Status.

3.1.1. Pet

Pet is the basic type in EasyPet language. It provides the user convenience to define a personalized pet in the program.

3.1.2 Status

Status is the part that makes ePet alive. Each status has a name that describe what the status is about. Each status has a status value which is automatically counted down toward 0 which mimics the gradual change of the corresponding status. User can set how fast the status changes and what can be done when the status value passes a threshold.

4. Basic Math

EasyPet language will provide add, subtraction basic math operations to handle the logic of status and attributes of Pet.

5. Sample Code

The sample code attached is an example of some EasyPet code that creates a simple electronic pet game. In this game, a pet named “jack” with one status “training”.

```
Panel {
   Status  {
      name: “training”,                  // This is the name of the status.

                  speed: 1,                  //  How much the status value decrease per second.
                                             //  The max value is 100, and it will be decreased
                                             //  according to the speed
                  value: 100,
                  action: ($OffTraining(name) 30 “Train”),      //  The user can take action as
                                                                //  described in OffTraining
                                                                //  function when the status
                                                                //  value falls below 30. The
                                                                //  action will appear to the
                                                                //  user as “Train”.
                  image: %1     // To set the first parameter in the status of pet as the current
                                //  image of the pet
                  sound: %2     // To set the second parameter in the status of pet as the
                                //  sound of the pet
   }

   Pet {
      type: “panda”
      name: “tuanyuan”,
      age: 1,
      height: 10.0,
      weight: 20.0,
      status: (“training” img1 sound1),   // the first parameter is used to specify which
                                          //  status will be called, and the second and the
                                          //  third parameter are used to describe the status
                                          //  of the pet
      image: “/src1”      // the picture to show when the pet is in its default,  perfect status
      }

   /* def means that the next block is the function which is defined by the user */
   def OffTraining(name) {
      while (name.value < 100)          // “this” refer to the status in which OffTraining is
      {
         r = random();
         if (r > 0.5) {
```

```
            name.value = name.value + 20;
            }
    }
}
```