HIGH DISTINCTION PROJECT

KUN MING CHUA s102858010

COS60006 Introduction to Programming

Introduction

This document will be helping new Ruby/TK developers to setup their workstations and introduce basic features of TK to create a basic read and write text that uses labels, buttons, and text inputs. Furthermore, this document will be listing down the pros and cons of using Ruby/TK for beginner projects. This document is created as a part of Introduction To Programming course as a high distinction project.

Introduction to Programming HD report video link:

https://www.youtube.com/watch?v=XX4xoc8Sqy0

Strength and Weakness

Strength:

- It improves the development time of GUI with their inbuild functions and features.
- It has numerous GUI that can be created using their library

Weakness:

- Outdated documentation for both installation and features.
- Some documentation examples resulted in errors instead of the implied features.
- Limited styling possible with their place, grid, and pack

Overall, this library offers a large variety of GUI and it improves our development time. However, it requires some time and dedication to navigate through their old documentations.

Tutorial Ruby/Tk

We will be creating a simple application that allows user to input a text file name and prints a label if the file exist. We will be using the pack geometry manager to manage the layout of our project. After this tutorial, you will be able to create a basic project with Ruby/TK, resize their start up window size, use labels/buttons/text fields, and updating GUI real time.

Step 1. Installing Ruby/TK

There are two main ways to install Ruby/TK. Tk was integrated within Ruby Installer on the previous years but was then removed after version 2.3.3.

“Download Ruby Installer 2.3.3”

URL: https://rubyinstaller.org/downloads/archives/

There are another way to install TK using the later versions of Ruby that requires a more elaborate setup and download size. For the purpose of the tutorial, we will be using the older version of ruby since it has all our required features.

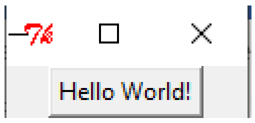
Step 2. Create a Project Folder and main.rb

After having Ruby/TK installed, let us create a project folder containing main.rb.

Let’s start by creating a simple hello world button to ensure that our Ruby/TK is properly installed

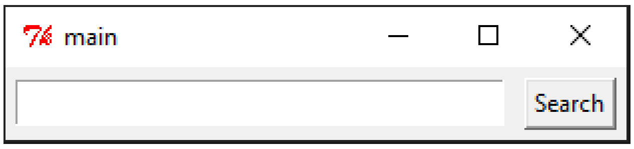
```ruby
require "tk"
root = TkRoot.new
button = TkButton.new(root) {
   text 'Hello World!'
   pack
}
button.configure('activebackground', 'blue')
Tk.mainloop
```

This code should create an additional window as shown below. This indicates that Ruby/TK is properly installed.



TkRoot can be represented as the window where the GUI elements will be attached on to. This root instance will be referenced during any GUI elements initialization such as TkButton. Finally, Tk.mainloop allows the whole project to run and its required on all tk projects.

Step 3. Create a Text field and Button



Let us start by creating our initial text field and button. We will need to indicate that our ruby script is using TK library.

require "tk"

Next, we need to initialize our Tk Root to be referenced by our future GUI elements and adding Tk.mainloop to allow tk to function.

Let us create a text field that is packed on the left side with 5 paddings on all sides.

```
text_field = TkText.new(root) do
    width 30
    height 1
    borderwidth 1
    font TkFont.new('times 12 bold')
    pack("side" => "left",  "padx"=> "5", "pady"=> "5")
 end
```

Then we can add a search button on the right side of the window.

```
button = TkButton.new(root) do
   text 'Search'
   pack("side"=>"right","padx"=> "5", "pady"=> "5")
end
button.configure('activebackground', 'blue')
```

Step 4. Add Button Functions

For this tutorial, we should make text_field and button as global variables.

```
$text_field = TkText.new(root) do
    width 30
    height 1
    borderwidth 1
    font TkFont.new('times 12 bold')
    pack("side" => "left",  "padx"=> "5", "pady"=> "5")
 end
$button = TkButton.new(root) do
   text 'Search'
   pack("side"=>"right","padx"=> "5", "pady"=> "5")
end
```

This will allow our functions to be able to access our variables which is required for this tutorial. For personal projects, it is recommended to use instance variables with classes for a more robust project.

Let us create a function that allows the user to look for a file written inside the textfield.

```
def findFile()
   temp = $text_field.get('0.0','end')
   puts temp
   if(File.file?(temp.chomp))
      puts "found it"
   else
      puts "does not exist"
   end
end
```

Ruby/TK TkText has a get function that takes 2 parameters to allow developers to grab the information inserted into it.

TkText.get(start,end)

We will be using the string received from our text field and search for the file in the same root folder as the script which will be our project folder.

To link the function with our button, we need to add an extra line to our TkButton called command. The script should look something like this:

```
$button = TkButton.new(root) do
   text 'Search'
   command(proc{findFile})
   pack("side"=>"right","padx"=> "5", "pady"=> "5")
end
```

This will trigger the function every time the button is pressed.

After finishing step 4, you will be able to right a file name and check if the file exists.

Step 5. Creating Labels Dynamically

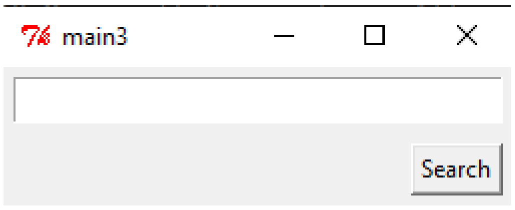
We will now create an additional label that shows our text file first line content. To accomplish this, we will need to do a basic file read and close.

```
file = File.new(file_name.chomp, "r")
file.close
```

Next, let us create a text label to print the content on the screen.

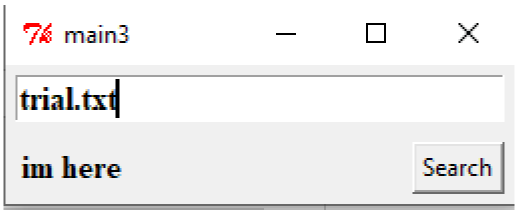
```
$label = TkLabel.new(root) do
        text file.gets.chomp
        font TkFont.new('times 12 bold')
        pack("side" => "left",  "padx"=> "5", "pady"=> "5")
     end
```

Finally, let us organize the position of the GUI elements to look like this.

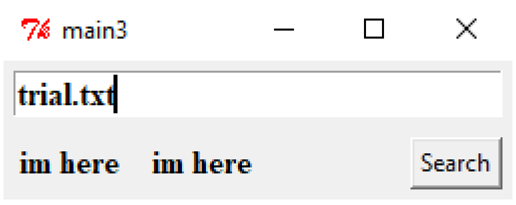


We should pack our textfield to top side and search button on the right side. By changing their pack "side" accordingly.

Create a random text file and insert a line in it. We can test our application and this should be how it looks.



All the core functionality is done, but there is still a problem of clicking search button multiple times which will cause the application to have several labels instead of one.



Next step will allow you to dynamically remove or hide GUI elements.

Step 6. Pack_Forget()

To dynamically remove or hide GUI element, we can use their forget feature.

pack_forget(), grid_forget(),place_forget()

Let us add a if statement that checks if label already exist, remove it from window.

```
if($label != nil)
        $label.pack_forget()
     end
```

And now we can use our application to read the first line of any text document in the same root folder as our code.

Conclusion

Ruby/TK is a great way to learn and setup GUI elements for our application and has great flexibility as shown in this tutorial. However, developers are required some basic understanding of their language to try and fix any potential errors in their example provided on their website. It is a library worth learning due to its functions and flexibility. To learn more, please visit

https://www.tutorialspoint.com/ruby/ruby_tk_guide.htm