Gooey Documentation

Release 1.0

Emily Johnston, Leah Cole, Kenny Suh, Mary McCreary, Evan Har

Jul 14, 2017

Contents

Contents:

CHAPTER 1

About

Hello and welcome to Gooey Docs!

What is Gooey?

Gooey is a tool for quick and easy creation of Graphical User Interfaces (GUIs). A GUI is the layout of an application - the buttons, the menus, the style, and so on. GUIs are notoriously troublesome to make, even for experienced programmers. The language was designed for people who have very little experience coding, and so is designed to look more like an English sentence than long blocks of code.

What are the Gooey Docs?

Gooey is meant to be easy and intuitive, but a tool is only as good as it's documentation. The Gooey team worked tirelessly to create flawless* documentation so that their users could create the very best GUIs possible. Here users can find language basics, tutorials, and all the nitty-gritty details of each object and attribute.

* some restrictions may apply.

About the Creators

Gooey was created as a comps project by a group of six senior Computer Science majors at Carleton College. They're swell people with a passion for puns.

CHAPTER 2

Getting Started

Language Basics

The Gooey GUI language was designed to be easy! Our goal was to make coding look less like code, and more like an English sentence. There are a few basic rules to follow, and then you'll be on your way.

Normally, each line starts with either **make** or **set**, which can be capital or lowercase.

As Gooey attempts to simulate English, each line of Gooey **ends with a period**. When making a list of multiple attributes, lines are **separated by commas**.

Although Gooey is similar to Python, in Gooey **white space does not matter**.

Make Commands

You can use the *make* command to create and add new elements to your GUI.

make commands always have the same basic syntax. Typing:

```
make Object name.
```

will create a new object of the given type with its default values.

Alternatively, the syntax:

```
make OBJECT NAME with
ATTRIBUTE value.
```

will create the object with most of the defaults, with one attribute modification that you specify.

You can add as many attributes as you want in the *make* command, by **separating the attributes with commas**. The syntax this time might look like:

```
make OBJECT NAME with
ATTRIBUTE VALUE,
ATTRIBUTE VALUE,
ATTRIBUTE VALUE.
```

The command *make* is always followed by the type of **object** you want to make. Types of objects always start with capital letters and might include:

- Window
- Button
- Menu
- Text
- Checkboxes

And so on.

Set Commands

If you choose to use the live Gooey editor, you might want to modify your objects after making them. *Set* commands allow you to change any attributes for objects you have already created.

The *set* command syntax is:

```
set NAME ATTRIBUTE VALUE.
```

You can also set multiple attributes of a single objects by using commas:

```
set NAME ATTRIBUTE VALUE, ATTRIBUTE VALUE, ATTRIBUTE VALUE.
```

Variable Names

After the object comes the **variable name**. Your object has to start with a lowercase letter, be only one word long, and contain only letters, numbers, or underscore _. Valid variable names include:

- `myvariable`
- `myVariable2`
- `mINE`
- `not_yours`
- `o_o`

These variable names are NOT valid:

- `my variable`
- `MyVariable`
- `mINE!`
- `\(^-^)/`

Attributes and Values

Attributes are a variety of customizable components of your object. Common attributes include:

- size
- color
- text
- position

Some objects may have additional unique attributes.

There are a lot of attributes where commonly desired **values** are preset for you, like the *size* attribute has default values of *small*, *medium*, and *large*. However, for your custom needs, you can also set your own size numerically. So, `size large` and `size 100 100` are both legal ways to define the size.

Syntax Example

In summary, to create a button, say:

```
make Button mybutton.
```

To create a button with the words "Yes" on it, type:

```
make Button mybutton with
text "Yes".
```

And to create a big button with the words "Yes", type:

```
make Button mybutton with
text "Yes",
size large.
```

Now you're ready to start making your own Gooey GUI!

CHAPTER 3

Tutorials

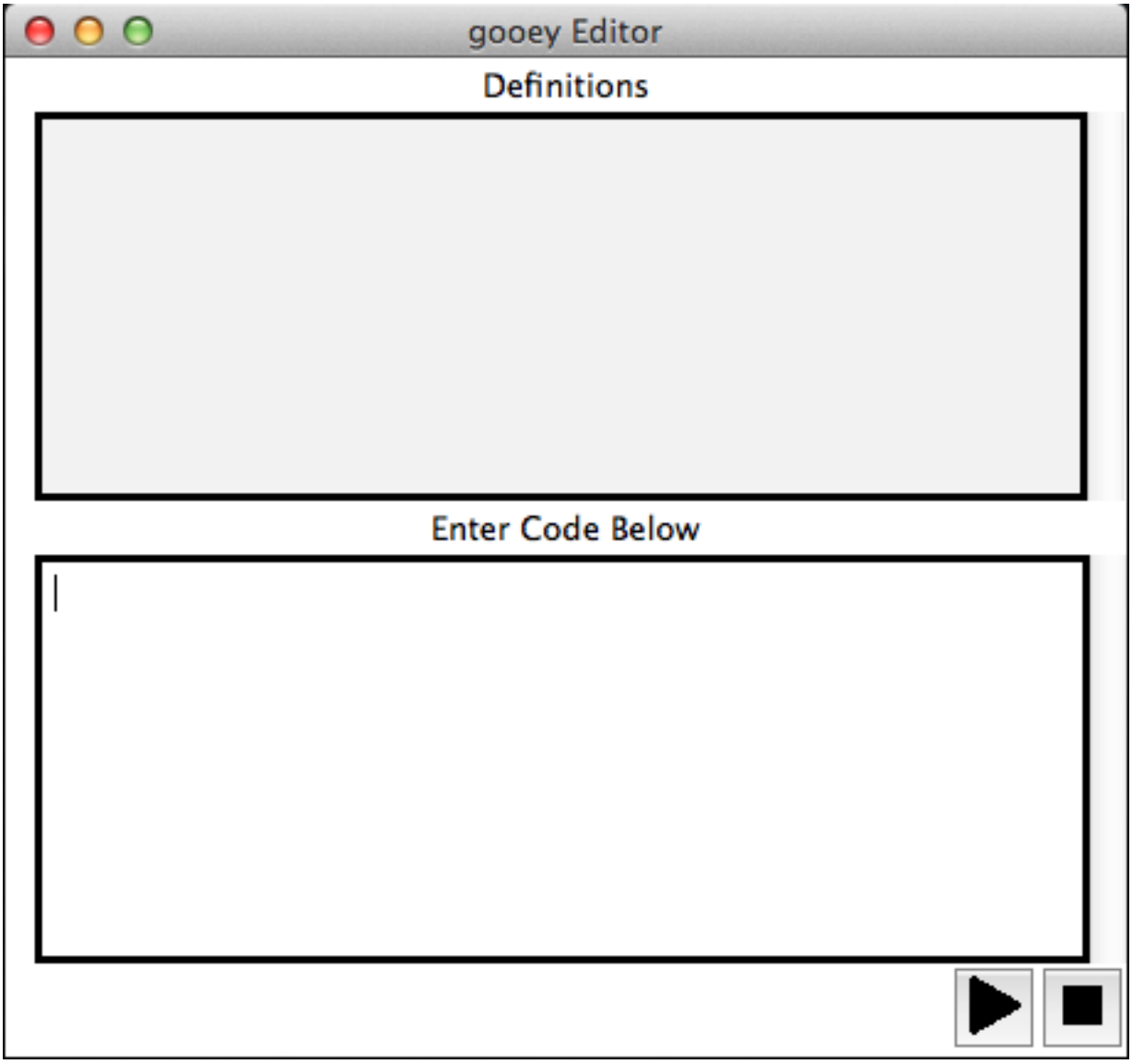
These tutorials are here to show examples of basic Gooey GUIs. If you're just starting to learn Gooey, you might want to go through some or all of the tutorials before you make your own GUI.

Window, Text, and Button Tutorial

To start using Gooey, go to the terminal and navigate to the project folder. Type in the terminal:

```
python3 gooey.py
```

Your Gooey editor will appear.

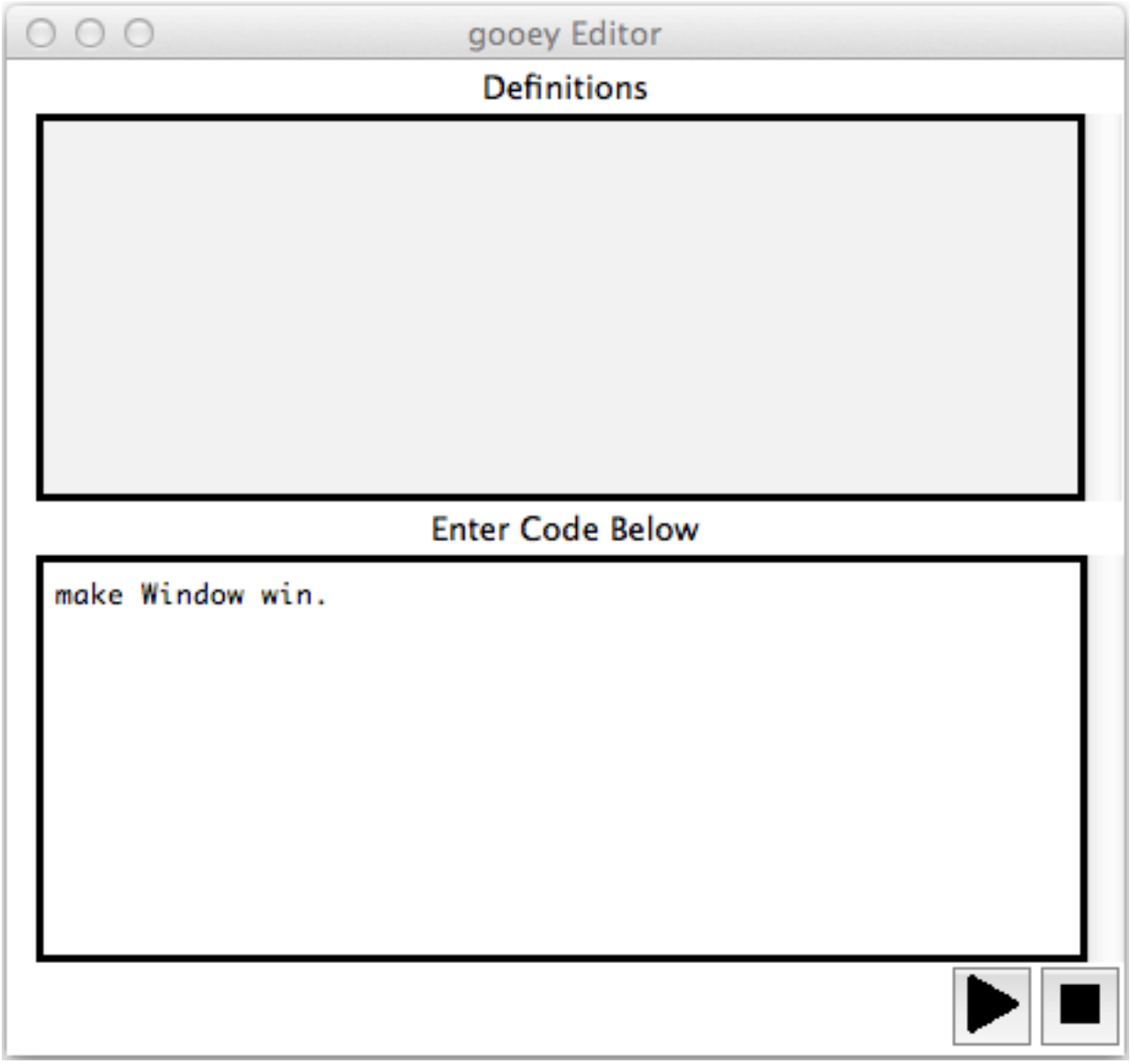


You can now enter the Gooey language into the editor.

To start, make an empty window. Type in the editor:

```
make Window win.
```

When you're ready to run your GUI, hit the play button at the bottom right corner.



This is your live preview. Right now it's a little boring.

Make the window bigger or smaller with the set command. The set command allows you to modify one or more things at a time. Try adjusting the title of the window, which will appear in the bar across the top. Type:

```
set win size 200 200, title "My First Gooey".
```

An empty window is useless though. To add writing to your GUI, you'll need to add a Text object. Type:

```
make Text greeting with text "Welcome to Gooey!".
```

Now try making a button. If you don't specify some attributes, Gooey will fill it in for you. Type:

```
make Button go with position left.
```

After you Run, your GUI should look like this

Your button is currently useless. You can click it, but since we haven't given it an *action* attribute, it won't do anything. You can modify existing objects further by using the *set* function. You can also write as many lines of Gooey code you want in the editor before you hit run.

Try:

```
set go text "Go", action windowColorChange green.
make Button stop with text "Stop", position right, action windowColorChange red.
```

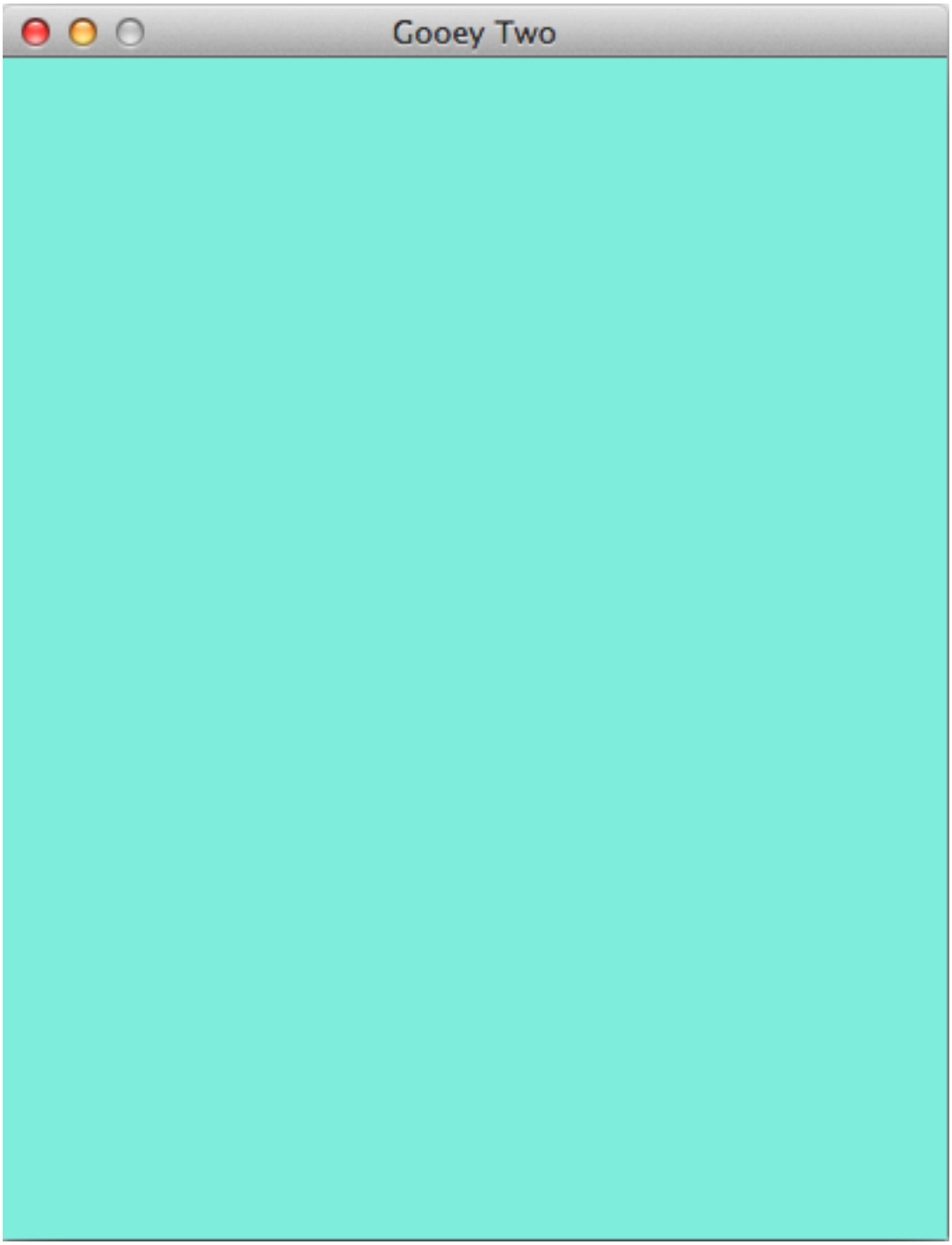
Try clicking on the button, see what happens. Gooey has a number of built in actions which you can find in the Actions section of the Gooey Docs.

FormattedText, TextBox, Checkboxes, and Function Tutorial

Now, let's try making a slightly more complicated Gooey.

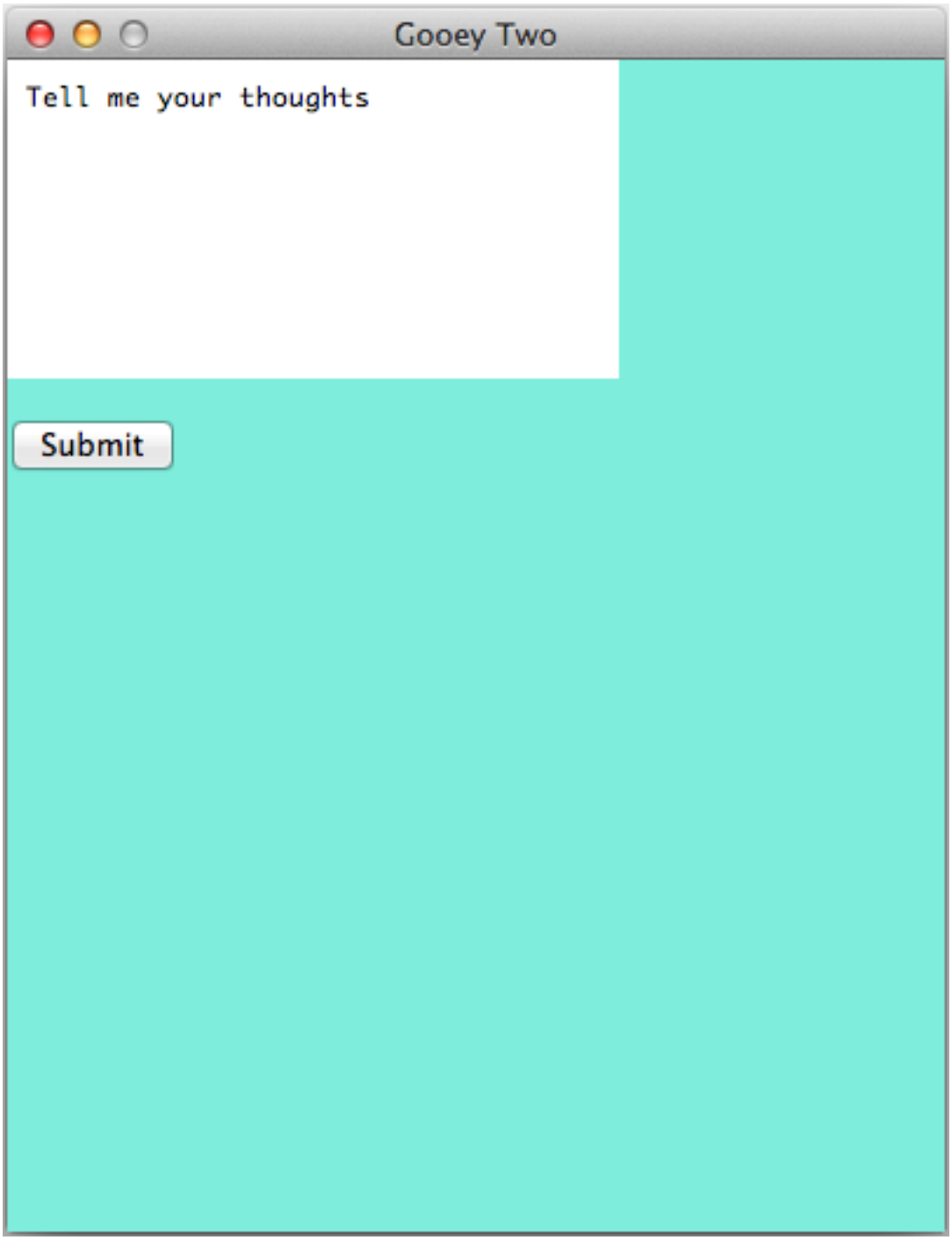
Remember, always start by making a window:

```
make Window win with title "Gooey Two", size 400 500, color #8fefdc.
```



There are times when you might want your users to write something out. A TextBox is perfect in this situation! Type:

```
make TextBox comments with text "Tell me your thoughts".
make Button submit with text "Submit", position 0 150.
```

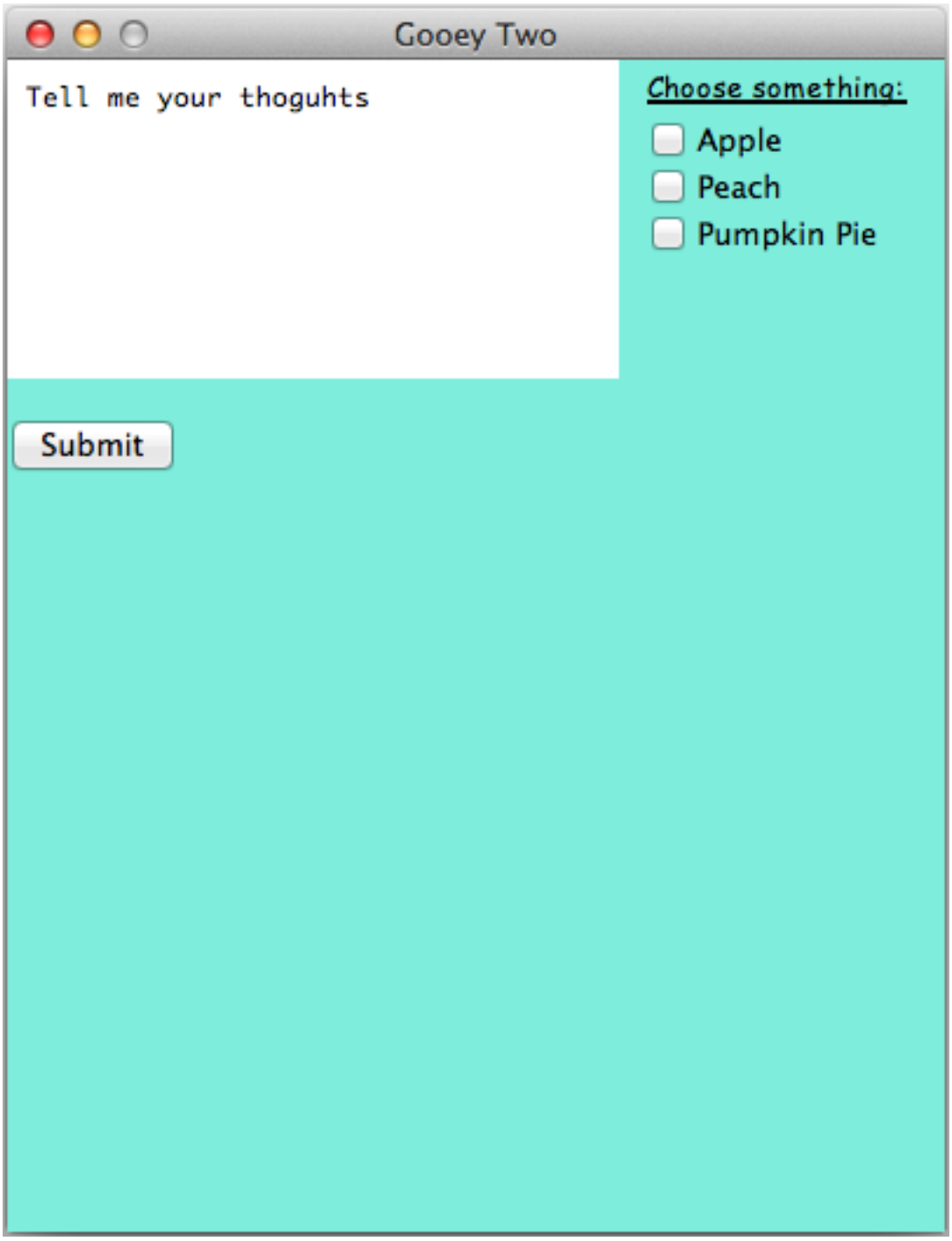


So far, everything looks the same. One way to diversify your interface to fit your asthetic is to use FormattedText options:

```
make FormattedText cbtitle with text "Choose something:", font "Comic Sans MS",␣
↪underline true.
```

You're going to have to wait to put that object somewhere. Maybe instead of direct input, you merely want your users to choose between a few choices. Try setting the title of some Checkboxes with your new FormattedText object. A simple Checkboxes object looks like:

```
make Checkboxes yummy with title cbtitle, options "Apple" "Peach" "Pumpkin Pie",␣
↪position 270 0.
```



Gooey only has a few default actions - lucky, users can make their own! A simple function to simply change the window color looks like this:

```
function myFunction(win) does set win color cyan.
```

After making a function, there are two ways to use it. To run myFucntion on the Window w, either you can say:

```
run myFunction(w).
```

or you can turn your function into an action for Buttons and MenuItems

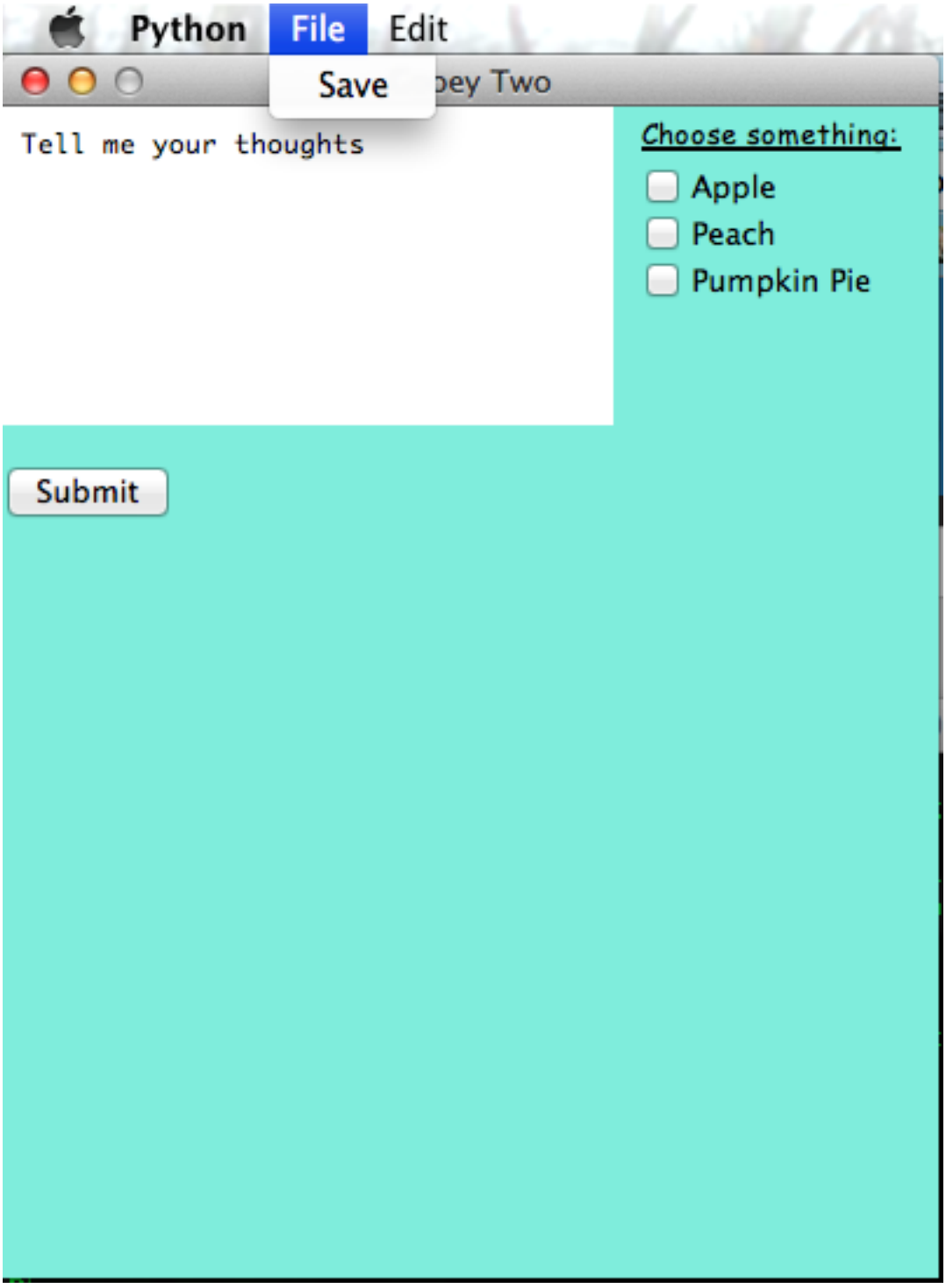
```
set submit action myFunction win.
```

Now for one of the most complicated parts of Gooey - menus. Menus actually use the *Menu* object and the *MenuItem* object. First, in double quotation marks list the name of the menu as you want it to appear across the top, a colon, and then the name of the MenuItem that will handle the drop down menu from that top level. For example:

```
make Menu m with menuoption "File":file "Edit":edit.
```

Alone, this will make a bar across the top that says File and Edit, but won't do anything. Make the MenuItem with the lower level options in the same format:

```
make MenuItem file with menuoption "Save":quit.
```



You may want to write all the code in a .txt file instead of dynamically generating it in the Gooey editor. You can rum those files by typing in the terminal:

```
python3 gooey.py input.txt
```

For everything in this tutorial, put the following into a .txt file:

```
make Window w with title "Gooey Two", size 400 500, color #8fefdc.
make TextBox comments with text "Tell me your thoughts".
function change(win) does set win color cyan.
make Button sumbit with text "Submit", position 0 150, action change w.
make FormattedText cbtitle with text "Choose something:", font "Comic Sans MS",␣
↪underline true.
make Menu m with menuoption "File":file "Edit":edit.
make MenuItem file with menuoption "Save":close.
```

Now you've gone through all the main components of Gooey. You're ready to make your own now!

CHAPTER 4

Actions and Functions

Actions and functions are extremely important elements of GUIs. Actions and functions make things happen.

Actions

Actions are the attributes for objects like *Buttons* and *MenuItems* that make them do things. For example:

```
make Button quit with title "Stop", action quit.
```

will make a button that shuts down the program when clicked.

quit is one of a handful of default actions Gooey has created for you.

Write

The *write* action will print a string to the terminal. After declaring the write action, add a string in double quotation marks, as follows:

```
make Button b with action write "I'm a print statement".
```

Quit

The *quit* action closes the program.

Example:

```
make Button b with action quit.
```

windowColorChange

The *windowColorChange* action changes the color of the window. After declaring the *windowColorChange*, add a color value.

Example:

```
make Button error with title "Error", action windowColorChange red.
```

Functions

A function will let you *create your own actions*. The syntax for functions is as follows:

```
function NAME(VARIABLE) does GOOEY_CODE; return VARIABLE.
```

A real example would look like this:

> function myFunction(win) does set win color green; return win.

This line of code means that the function named myFunction, which takes in an object *win*, will reset the window color to green. You need to return the object at the end for your changes to occur.

There are two ways to run your function. You can use the *run* command to run the function yourself:

```
make Window w.
function myFunction(win) does set win color green.
run myFunction(w).
```

Alternatively, you can set the function as a Button.:

```
make Window w.
function myFunction(win) does set win color green; return win.
make Button b with action myFunction w.
```

CHAPTER 5

Attributes

Attributes control the details of each Object you create. Not all Objects can have the same attributes, and some attributes are specific to one or two Objects.

Each attribute needs a *value*. Some attributes accept values in multiple different formats.

Title

A title generally creates text *above* the Object.

In a **window**, a title will put a name at the very top bar across the application window.

Values:

- A plaintext string

Text

Text will generally create words on top of or inside the object.

Values:

- A plaintext string

Options

Options will generally be a list describing the different Checkboxes, RadioButtions, or Menu/MenuItems.

Values:

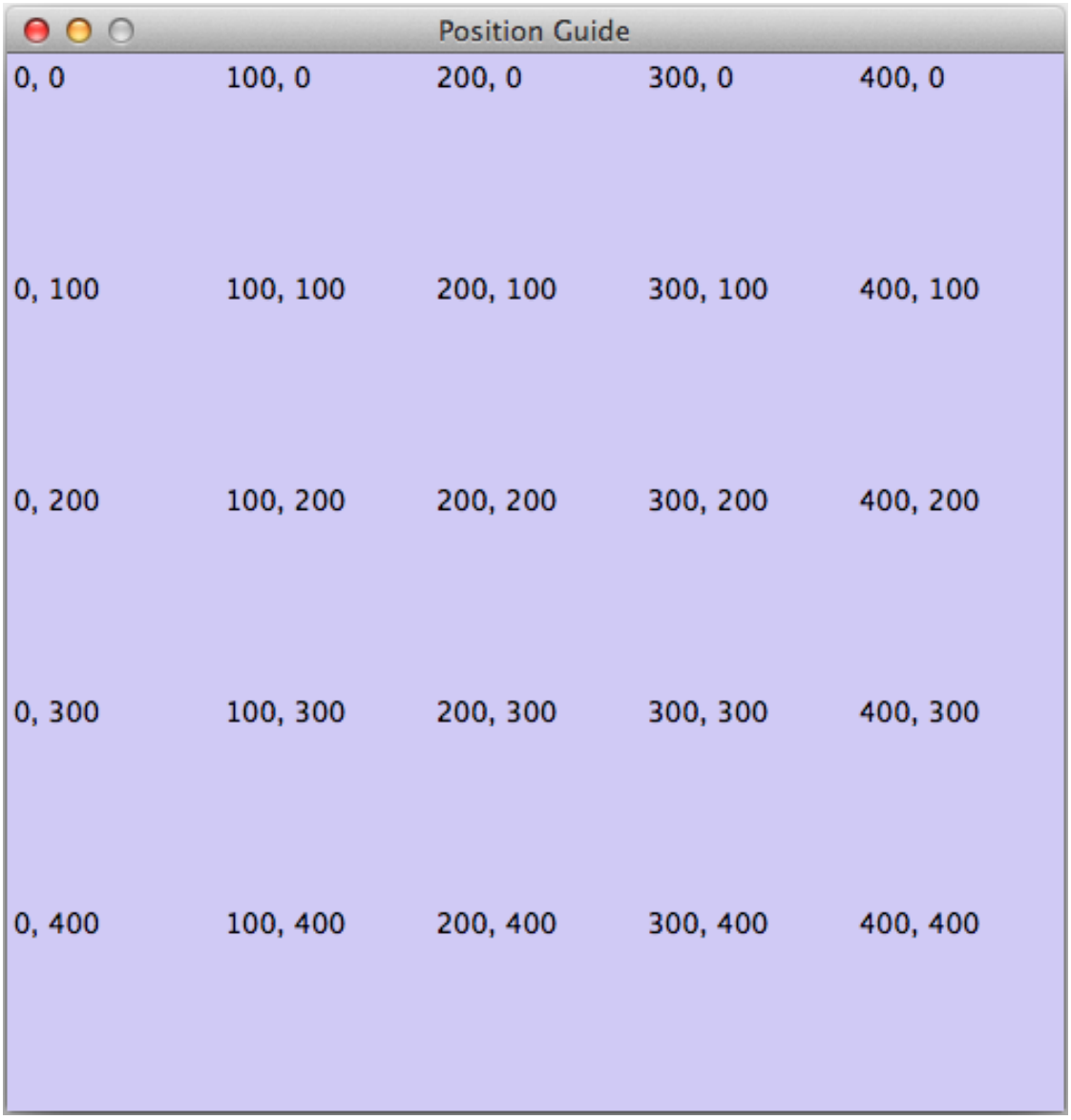
- **A space separated list of plaintext strings surrounded by double quotation marks.**
  - Example: “Option1” “Option2” “Option3”

- **For *Checkboxes*, an * before one or more of the strings will create those values automatically checked**
  - Example: *"Option1" "Option2" *"Option3"
- **For *RadioButtons* an * before one of the strings will create that value automatically selected**
  - Example "Option1" *"Option2" "Option3"

Position

Position allows you to move your objects around the window. If you want to set the position of an Object, you must first set the size of the of the Window. Position can be set as an x y coordinate by **pixels**, with x an y as plain integers seperated by a space.

For an idea of pixel positions, refer to the following *size 500 500* window.



Values:

- **A position keyword**
  - center
  - top
  - bottom
  - left
  - right
  - topleft
  - topright
  - bottomleft
  - bottomright
- **Height and width integer coordinates, separated by a space.**
  - Example: position 3 5

Size

Size allows for the adjustment of the height and width of the object.

Values:

- **A size keyword**
  - small
  - medium
  - large
- **Height and Width integers, separated by a space.**
  - Example: size 2 5

Color

Color changes the color of the object.

Values:

- **A color keyword**
  - red
  - blue
  - yellow
  - orange
  - green
  - purple
  - pink
  - cyan

  - magenta
  - white
  - black
- **A RGB value, separated by spaces.**
  - Example: color 000 000 255.
- **A hex value**
  - Example: color #33aa00

Action

Action sets the events that happen after an object is interacted with.

Values:

- **A built-in Gooey action:**
  - quit
  - write
  - changeWindowColor
  - changeWindowSize
- **A custom function:**
  - See: Functions and Actions

Hidden

Determines if object can be seen. Default is **always false**, object is *not* hidden. Setting hidden to true will hide the Object without destroying it.

Values:

- true
- false

Font

Changes font used if Object incorporates text. Default is "Times New Roman"

Values:

- **A font name, surrounded by double quotation marks.**
  - "Times New Roman"

Size (for FormattedText)

Changes font size for FormattedText object

Values:

- An integer for font pt size

Bold

Changes text to bold font when set to True. Default is **False**.

Values:

- true
- false

Italic

Changes text to italicized font when set to True. Default is **False**.

Values:

- true
- false

Underline

Changes text to be underlined when set to True. Default is **False**.

Values:

- true
- false

Source

The path or filename for the Image object. Source files **must be in .gif format**

Values:

- **filename surrounded by double quotation marks**
  - Example: source "images/apple.gif"

CHAPTER 6

Objects

Objects are the building blocks of your GUI. Each object has a different set of uses, attributes, and functions.

Button

A button is a clickable object.

Example syntax:

```
make Button b.
make Button b with text "Hello".
make Button b with text "Hello", position 100 25.
set b action close.
```

Attribute	Description	Possible Values	Default Value
text	Words on button	• A plaintext string	"Untitled Button"
position	location of button on window	• position keyword • integer pixels, separated by space	center
size	size of button	• size keyword • width and height pixel integers, separated by a space	medium
action	Effect when button clicked	• name of Python or Gooey function	none
hidden	Determines if button visible (false) or invisible (true)	• true • false	false

Window

A window is the frame on which you create your GUI. The window is **always the first object you make**.

Example syntax:

```
make Window w with size 500 500, color green.
set w title "My favorite GUI".
```

Attributes:

Attribute	Description	Possible Values	Default Value
title	The name as displayed in the top bar of the window.	• A plaintext string	"Untitled Window"
size	The width and height of the window	• size keyword • width and height pixel integers, separated by a space	medium
color	The color of the window background	• color keyword • rgb value, separated by spaces (0 - 255) • #hexvalue	white
action	The effect from interacting with a window.	• name of Python of Gooey function	none
font	The font for all text used in the window	• String name of available font	Times New Roman
fontSize	Size of text	• Integer	12
textColor	Color of text	• color keyword • rgb value, separated by spaces	black

Checkboxes

Checkboxes are square boxes the user can click on to select any number of options. If you create a Checkboxes object without the *options* attribute, it will have three default checkboxes labeled "Option 1", "Option 2", and "Option 3". Placing an asterisk before any of the attributes will mark that option to be selected by default.

Example syntax:

```
make Checkboxes c with options "hello" "yellow" "fellow".
set c position 20 20.
```

Attribute	Description	Possible Values	Default Value
title	text above checkbox set	• A plaintext string	"Untitled Checkboxes"
options	The Checkboxes labels.	• strings in double quotes, separated by a space • string preceded by * to mark default selections	*"Option 1" "Option 2" "Option 3"
position	location of checkbox set in window	• integer pixels, separated by by space	center

RadioButtons

RadioButtons are circular buttons a user can click to select **one** option out of many. If you create a RadioButtons object without the *options* attribute, it will have three default buttons labeled "Option 1", "Option 2", and "Option 3". Placing an asterisk before one of the attributes will make that option to be selected by default.

Example syntax:

```
make RadioButtons r with options "hello" "mello" "jello".
set r title "Choose one:".
```

Attribute	Description	Possible Values	Default Value
title	text above RadioButtons set	• A plaintext string	"Untitled RadioButtons"
options	The RadioButtons labels.	• strings in double quotes, separated by a space • string preceded by * to mark default selected	*"Option 1" "Option 2" "Option 3"
position	location of RadioButtons set in window	• position keyword • integer pixels, separated by by space	center

Text

Text is a simple text region the user *cannot* interact with.

Example syntax:

```
make Text t with text "Welcome to Gooey! Please leave your shoes at the door."
set t color blue.
```

Attribute	Description	Possible Values	Default Value
text	unmutable words in a window	• A plaintext string	"Text"
position	location of text in window	• position keyword • integer pixels, separated by by space	center
size	size of text	• size keyword • width and height integers, separated by space	medium
color	color of text	• color keyword • rgb value, separated by spaces (0 - 255) • #hexvalue	black
hidden	Determines if text visible (false) or invisible (true)	• true • false	false

TextBox

TextBox objects create a space where users can type. When you create a TextBox with a *text* attribute, the value entered will appear as default text within the text box.

When setting the *size* of the TextBox using integers for width and height, the integers will set the width and height by **character count**. For example, size 15 10 will create a TextBox 15 *characters* across, with ten *lines* of height.

Example syntax:

```
make TextBox tb with text "Write your answer here".
set tb size large.
```

Attribute	Description	Possible Values	Default Value
text	mutable words within the TextBox	• A plaintext string	"Type Here"
position	location of TextBox in window	• position keyword • integer pixels, separated by by space	center
size	size of TextBox	• size keyword • width and height pixel integers, separated by space	medium
hidden	Determines if TextBox visible (false) or invisible (true)	• true • false	false

Menu

Menus are a list of actions. Menu's are created with Menu Items. When creating a Menu, the *menuoption* attribute points to the MenuItems to be included in the Menu. First, in double quotation marks list the name of the menu as you want it to appear across the top, a colon, and then the name of the MenuItem that will handle the drop down menu from that top level. A Menu *must* include MenuItems.

Example syntax:

```
make Menu m with menuoption "File":file "Edit":edit.
```

Attribute	Description	Possible Values	Default Value
menuoptions	The top level menu labels	• list of MenuItem objects separated by spaces	menuItem1 menuItem2 menuItem3

MenuItem

MenuItems are the terminal actions in a Menu. The variable name of the MenuItem must match the name of the correlating option listed in the Menu object. With the *options* attribute, MenuItems have two parts. First the text the user will select, then a colon, followed by the action or function.

Example syntax:

```
make MenuItem file with menuoption "Save":save "Quit":close.
```

Attribute	Description	Possible Values	Default Value
title	name visible in menu	• A plaintext string	"Untitled MenuItem"
menuoptions	The selections within the menu	• a MenuItem object • a terminal in the format "name":action	"Option1" "Option2" "Option3"

Image

Images are pictures you can add your your Gooey. The image must be in **.gif format** although the movement will not be maintained.

Example syntax:

```
make Image i with title "Apple", text "This is my most favorite apple", source
↪"images/apple.gif".
```

Attribute	Description	Possible Values	Default Value
hidden	Determines if Image visible (false) or invisible (true)	• true • false	false
source	path or filename of Image	• image file in .gif format	defaultIcon

FormattedText

FormattedText is an object that stores values for a text with different formatting options. It is first created by the user with attributes of their choice and then can be used in button text, checkboxes/radiobutton titles, and text objects.

Example syntax:

```
make FormattedText t with text "Hello World!", font "Arial", size 15, color blue,␣
↪bold true, italic true, underline true.
make Button b with text t.
make Checkboxes c with title t, options "Yay" "Nay", position 50 50, size medium.
```

Attribute	Description	Possible Values	Default Value
text	text to be stored in object	• A plaintext string	"Untitled Text"
font	font of text	• A plaintext string: "Times", "Arial"/"Helvetica", "Courier" "Comic Sans MS", "MS Sans Serif" "MS Serif", "Verdana"	"Times"
color	color of text	• color keyword • rgb value, separated by spaces (0 - 255) • #hexvalue	black
size	size of text	• integer (pt size)	12
bold	Determines if the text is bold (true) or not (false)	• true • false	false
italic	Determines if the text is italicized (true) or not (false)	• true • false	false
underline	Determines if the text is underlined (true) or not (false)	• true • false	false

CHAPTER 7

Indices and tables

- genindex
- modindex
- search