Paedia's Place

A Programming Portal to Procedural Perspicuity

Contents

1 Introduction

Abelson and Sussman on PROCEDURAL PROGRAMMING

The computer revolution is a revolution in the way we think and in the way we express what we think. The essence of this change is the emergence of what might best be called procedural epistemology - the study of the structure of knowledge from an imperative point of view, as opposed to the more declarative point of view taken by classical mathematical subjects.

1.1 What's it all about?

...

Hard Fun and the Pleasure of Programming

Fun isn't always hard. Sometimes it is frivolous. Programming isn't always pleasurable. Sometimes it is torturous. Yet the best fun, some say is hard fun. Hard fun is a term coined by Alan Kay to describe an activity that is both challenging and enjoyable. Hard fun, moreover, is a term that is quite often associated with the best learning experiences. And the best programming? Most hackers would agree that it is pleasurable programming in a very deep sense, in a sense best described by the word **flow**.

Flow is a term coined by Mihaly Csikszentmihalyi which refers to the mental state in which a person performing an activity is fully immersed in the activity, and derives enjoyment from the activity as a result of their energized focus and complete involvement in the activity. Flow is characterized by the complete absorption in what one does to the extent that one loses their sense of time.

What do the hard fun of learning and the pleasure of programming have in common? Flow! This text is intended to afford you an opportunity to find flow, at least to some extent, in the hard fun of programming in a very simple environment. As a side effect, you will gain learn a little bit about what it is like to think like a computer scientists, and enhance your intuitions about the potential of computational modeling to inform theories of cognition.

Powerful Ideas

I have always been attracted to the notion of "powerful ideas" as I have read the writings of Alan Kay, Marvin Minsky, and Seymour Paper. Yet there appears to be no pithy single definition seems to render this two word phrase meaningful. I certainly have no interest in taking a stab at one. But since I like to think that this text incorporates powerful ideas in the service of programming and thinking about programming, I feel that I should say a little something about what comes to my mind when I think on the phrase. For me, an idea is powerful if it can be described in a relatively short sentence. For me, an idea is powerful if it is applicable to a wide variety of situations. For me, an idea is powerful if its successful application feels just right, and makes you feel good about the thought process in which you are engaged. Different people will favor different powerful ideas, and their thinking will be informed by the palette of powerful ideas they bring to their work, and the system of administration that they have developed for deploying the powerful ideas at their disposal. The problem solving strategy of "problem decomposition" is a powerful idea for which nearly everyone has an intuitive feel. The principle of "invariance" is a powerful idea which affords the reuse of processes. These and other powerful ideas will be detailed as they are incorporated into the adventure in computer programming that is afforded by this text.

1.2 The Featured Software: Gargoyle, the Dotsworld, and Clay

The programming environment featured in this text is called `Gargoyle`. The limited portion of Gargyole that will be the focus of our attention in this text is loosely referred to as the `Dotsworld`. The actual language in which computations are expressed is called `Clay`.

The Nature of the Software

The software featured in this text is minimalist. Minimalism is the idea of focussing on a small set of things, and removing everything else that distracts from a focus on those things. Please note that there is no one minimalism in any realm, whether, say, software or painting or composing or cooking. Steve Reich and Philip Glass, for example, privedge very different notions (phasing vs add/drop operators) in their minimalist compositions. The form of minimalist software incorporated into this text features procedural abstraction and a most simple sort of repetition. Other constructs may be more important for a variety of practical and theoretical purposes, including parameter passing mechanisms, class constructions, and mapping functions. But the minimalism inherent in Clay, serves to enhance computational intuitions without requiring much by way of investment with concerns of syntax and semantics.

Downloading the Software

To download the `Gargoyle` program which houses the "Dot World", aka Paedia's Place, and nest it in a place that will work well for what you are going to be invited to do in this text, simply:

1. Create a place for it to nest by doing the following: (1) Make a folder called `Paedia` in a convenient location, and (2) Make a folder called `software` within the `Paedia` folder.
2. Find your way to: `www.cs.oswego.edu/∼blue/Paedia`
3. Download the file `Gargoyle.jar` to the `software` folder that you just made.

Running the Software

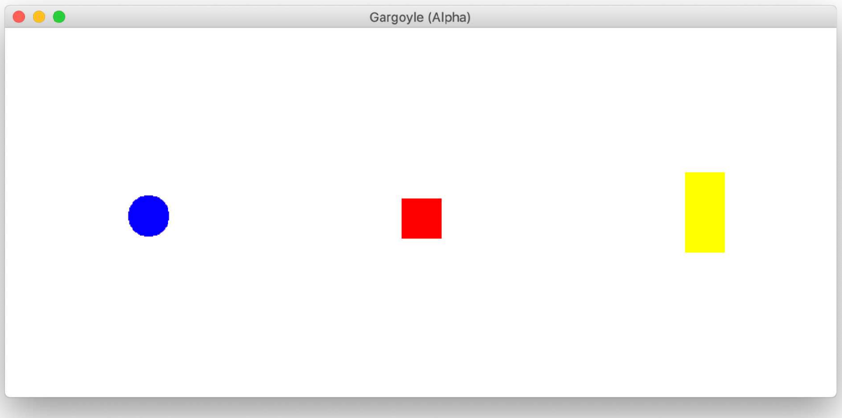
If you want to work with this software, you should launch this program from the command line of a terminal window on whatever machine you are working on. Once you have place the software in your `software` folder (see previous instructions), and once you have changed to the `software` folder in your terminal window, you need to type the following command:

```
java -jar Gargoyle.jar
```

Please note that, unlike the Google search box, for example, the command line interpreter of the terminal window is unforgiving with respect to spelling, case, and spacing! If you want to write programs, you will have to accept the fact that you generally speaking must attend to these sorts of details.

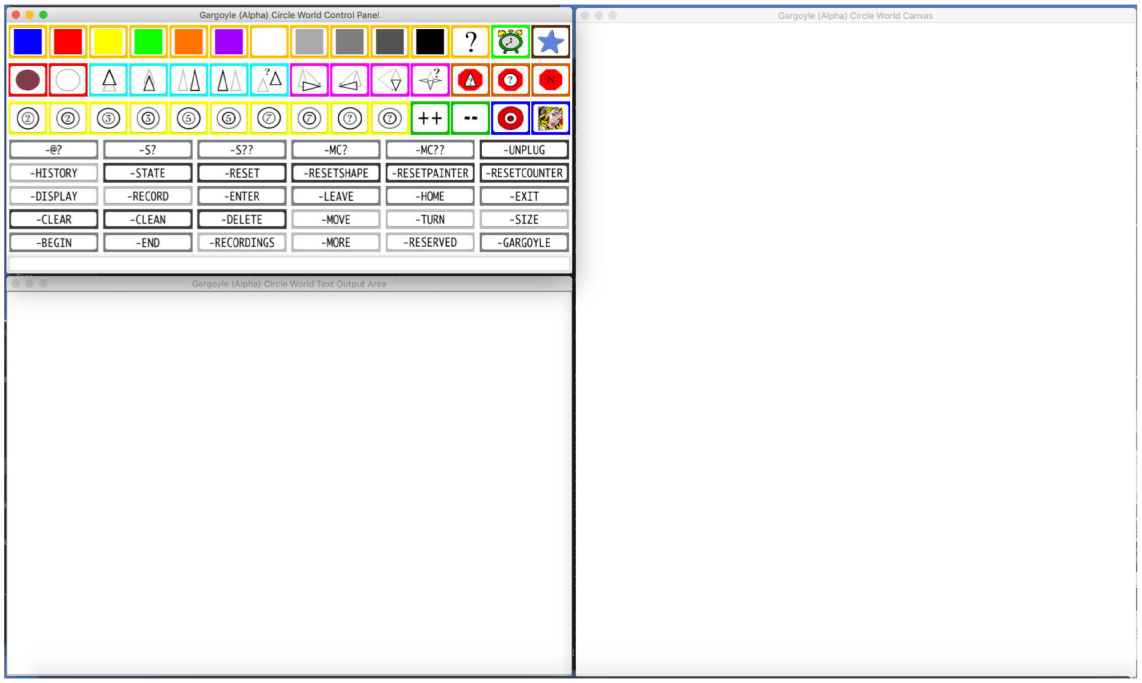
Finding Your Way to the Dotsworld

If all goes well, you should see an image like that shown below appear on your screen. To reach the Dotsworld, simply click on the dot. (The parallel worlds have been disabled.) The infrastructure for the Dotsworld will then appear on your screen.



Meeting the Dotsworld Infrastructure

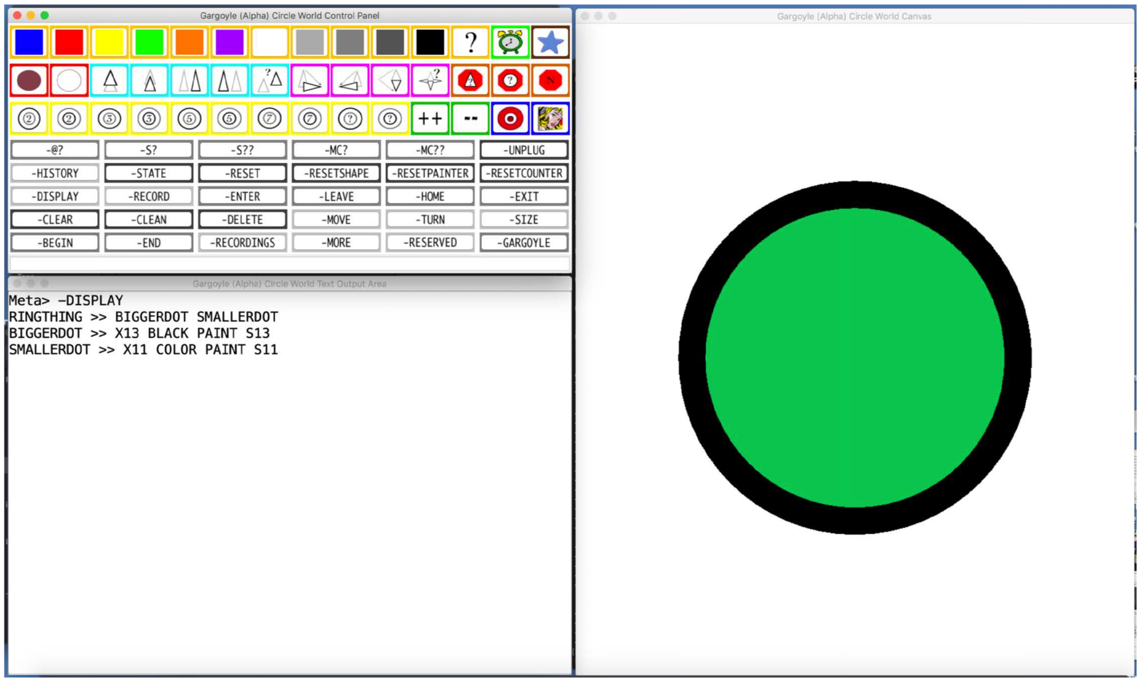
The visible infrastructure for the dots world is composed of three windows, a **control panel** in the upper left part of your screen, a **text output window** in the lower left portion of your screen, and a graphical output window, or **canvas** on the right of your screen, as shown below.



The text will walk your through interactions with the control panel in due course. For now, you might just enjoy letting your eyse wonder over the panel, casually thinking about what functionality might lurk within as you do.

Example Program and its Output

By way of wheting your appetite for things to come, you might like to take a look at the following image which contains a listing of a program in the text output window and the image that was generated when the program was run in the canvas. The three line program was entered one line at a time in the text input box which appears at the bottom of the control panel. I then ran the program by typing its name, `RINGTHTHING` in the text input box (not shown in the image). Finally, the program was displayed by clicking on the `-DISPLAY` button and typing the name of the program `RINGTHING` into a widget that appeared on the screen.



```
Meta> -DISPLAY
RINGTHING >> BIGGERDOT SMALLERDOT
BIGGERDOT >> X13 BLACK PAINT S13
SMALLERDOT >> X11 COLOR PAINT S11
```

1.3 Video Invitation and Web Site Anticipation

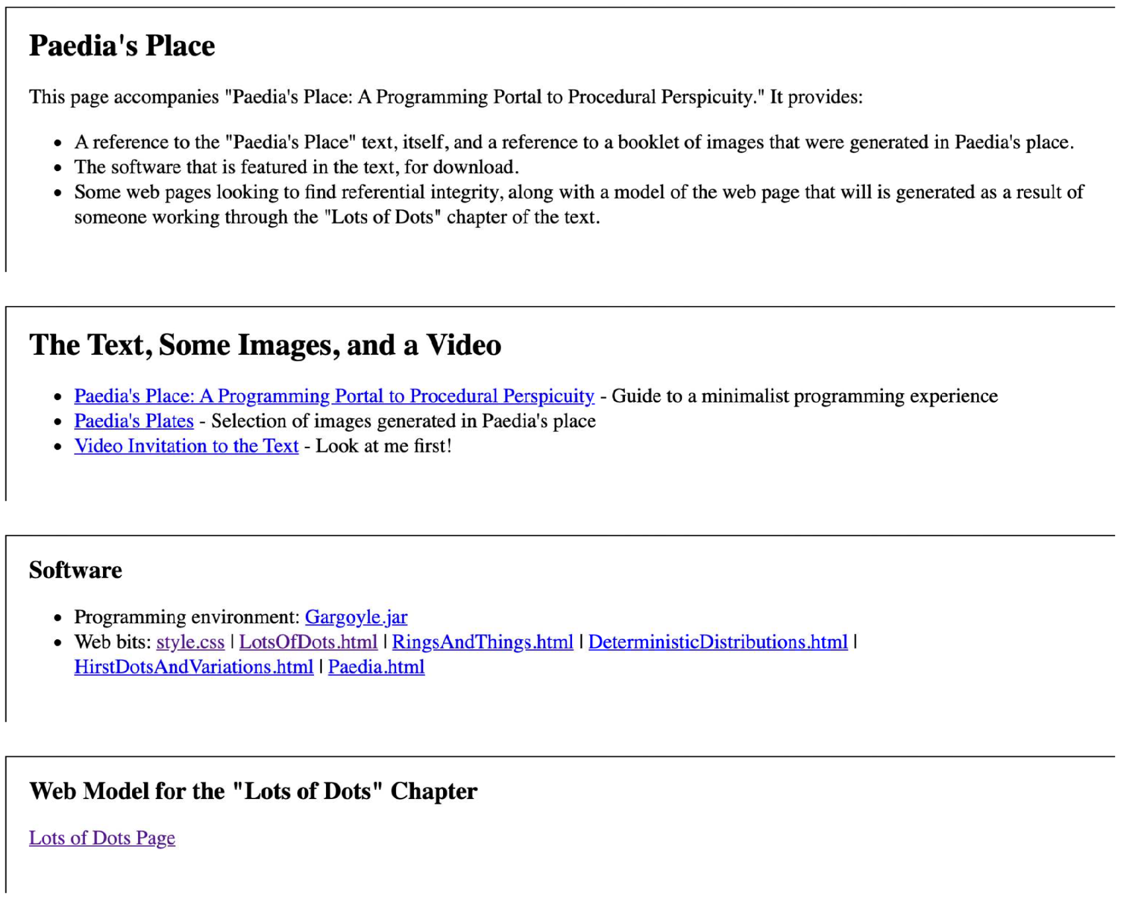
Beginnings can be awkward. Endings can be satsifying. These statements hold for dancing a tango, visiting a foreign land, or engaging in a computer programming experience. A video invitation is presented in order to assure a relatively smooth start to the programming experience in which you are about to engage. An image of a web page is then presented in order to foreshadow an artifact that will result if you consciensously work through the exercises in this text, a small web site that will reflect in a particularly visual way the work that you will have completed by working through the text, and which could very well leave you with at least a modest feeling of accomplishment.

The Video Invitation

...

The Web Site Preview

...



Paedia's Place

This page accompanies "Paedia's Place: A Programming Portal to Procedural Perspicuity." It provides:

- A reference to the "Paedia's Place" text, itself, and a reference to a booklet of images that were generated in Paedia's place.
- The software that is featured in the text, for download.
- Some web pages looking to find referential integrity, along with a model of the web page that will is generated as a result of someone working through the "Lots of Dots" chapter of the text.

The Text, Some Images, and a Video

- Paedia's Place: A Programming Portal to Procedural Perspicuity - Guide to a minimalist programming experience
- Paedia's Plates - Selection of images generated in Paedia's place
- Video Invitation to the Text - Look at me first!

Software

- Programming environment: Gargoyle.jar
- Web bits: style.css | LotsOfDots.html | RingsAndThings.html | DeterministicDistributions.html | HirstDotsAndVariations.html | Paedia.html

Web Model for the "Lots of Dots" Chapter

Lots of Dots Page

2 Lots of Dots

Marvin Minsky on MICROWORLDS AND LEARNING

In doing this (working in the world of children's blocks), we'll try to imitate how Galileo and Newton learned so much by studying the simplest kinds of pendulums and weights, mirrors and prisms. Our study of how to build with blocks will be like focusing a microscope on the simplest objects we can find, to open up a great and unexpected universe. It is the same reason why so many biologists today devote more attention to tiny germs and viruses than to magnificent lions and tigers. For me and a whole generation of students, the world of work with children's blocks has been the prism and the pendulum for studying intelligence. *In science, one can learn the most by studying what seems the least.*

2.1 Scattered Black Dots

Two Clay sessions introduce some primitive Clay programs, the mechanism for defining new Clay programs, the sole repetition construct that will be used in this text, and a number of Meta commands that serve a range of useful infrastructural purposes.

Text Segment #1: First Clay Experience in the Dotsworld

Recall that the two principal objects lurking about in Dotsworld are the painter and the circle, neither of which has a name. So long as there is only one of a (kind of a) thing, you can easily get by without naming the thing!

The first line in Clay Session #1 consists of the `PAINT` command, which simply asks the unnamed painter to paint the unnamed dot, right where the painter happens to be. It is presumed that the painter is somewhere in Dotsworld, that the circle has a radius, and the some color has been selected by the painter. Just for the record, the default values for these properties are: location of the painter is the center of the world, size of the circle is radius=20, and the color selected is black. The result of the painter executing this command can be seen in Figure #1 (a).

The second line in Clay Session #1 consists of the `MOVE PAINT` command sequence, which asks the unnamed painter to move to a random location in the canvas (the `MOVE` command), and then to paint the dot at that new location (the `PAINT` command). The third line is identical to the second, and causes another dot to be rendered. Figure #1 (b) represents the state of the canvas after the third line was executed.

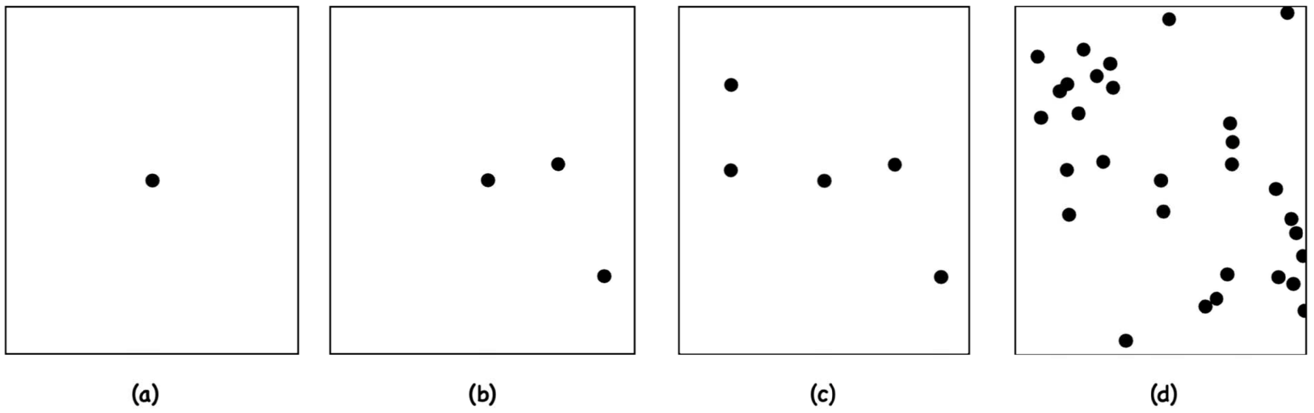
The fourth line in Clay Session #1 serves to **define** a new program, named `DOT`, which denotes the `MOVE PAINT` command sequence. Note that, consistent with a pathological interest of the Clay language designer to let certain tokens breath, the command definition symbol `>>` must be surrounded by spaces. Each of the next two lines **use** the `DOT` command. The result of twice executing the `DOT` command is seen in two more dots appearing on the canvas. Figjure #1 (c) which asks the unnamed painter to move to a random location in the canvas (the `MOVE` command), and then to paint the dot at that new location (the `PAINT` command). The third line is identical to the second, and causes another dot to be rendered. Figure #1 (b) represents the state of the canvas after the sixth line was executed.

The seventh and final line of Clay Session #1 illustrates the only **repetition** construct that will be employed in this text. You can repeat a command simply by prepending a positive integer to the command (note that there is no intervening space). Thus, the construct on this line results in the rendering of 25 additional randomly located dots. Please see Figure #1 (d) for the state of the canvas after Clay Session #1.

Clay Session #1: Dotting the World

```
Clay> PAINT
Clay> MOVE PAINT
Clay> MOVE PAINT
Clay> DOT >> MOVE PAINT
Clay> DOT
Clay> DOT
Clay> 25DOT
```

Figure #1: Dotting the World



Text Segment #2: Scattered Black Dots Program and Some Meta Commands

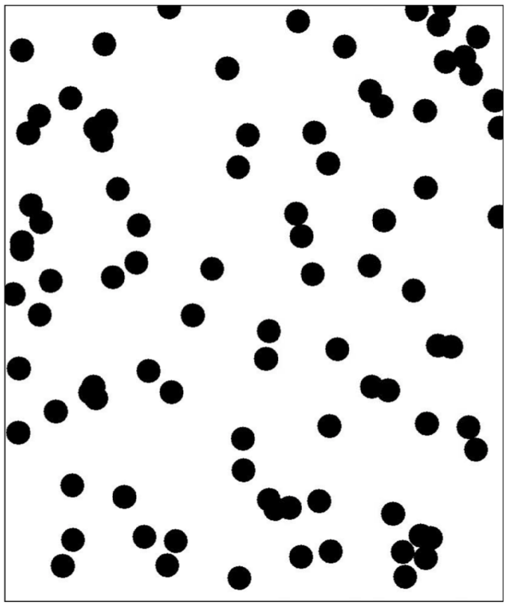
There are Clay commands and there are Meta commananands in the featured programming system. Clay commands operate on the objects at hand, most notably the painter and the circle. Meta commands perform useful infrastructural operators with respect to the system. I tend to reference the meta commands by prepending a dash to them. The `-CLEAN` meta command essentially erases the canvas. The `-RESET` meta command reinitializes the values of most properties to their default values. For example, it repositions the painter so that it will be found in the center of the screen, regardless of where it might have been just prior to issuing the meta command. The `-RECORD` command prompts, in a widget that pops up on the screen, for a file name, and then creates a file with that name containing an image of the canvas, reporting on just where it puts it. The `-DISPLAY` meta command prompts, in a widget that pops up on the screen, for the name of a program that you want to display, and it lists the program, together with any subprograms that it calls, together with any subprograms called by any subprograms, and so on. Please take a look at Clay Session #2, and see if you can wrap your mind around all of it.

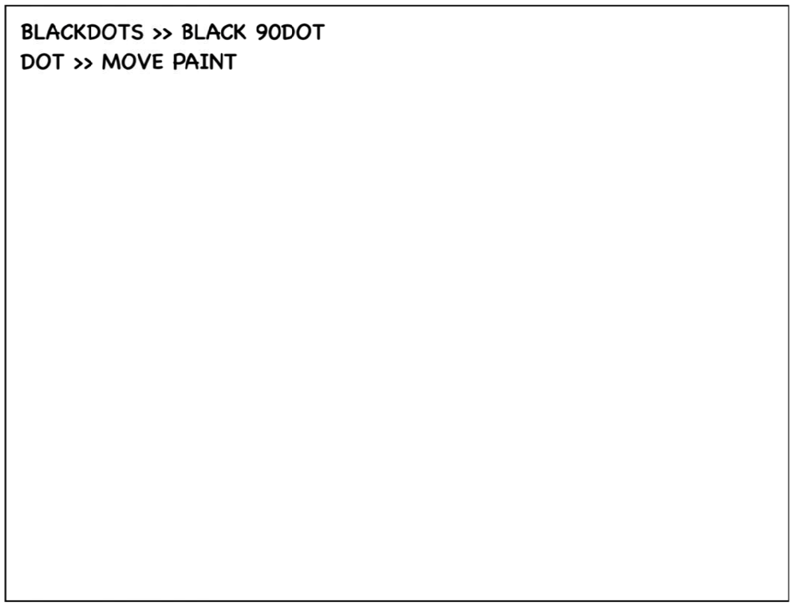
I generally click on buttons in the control panel corresponsing to meta commands in order to run them. The first line of this session represents the fact that I clicked on the `-CLEAN` button in order to run the meta command to erase the canvas. The second line of this session represents the fact that I clicked on the `-RESET` button in order to run the meta command to reset the values of the properties of the painter and the circle to their default values. The third line constitutes the definition of a command called `BLACKDOTS` to draw 90 randomly located black dots of size equal to the circle's default size. The fourth line runs the command. Next I arrange for the image in the canvas to be saved to a file by clicking on the `-RECORD` button and running the associated meta command. Lastly, I arrange for a reasonable listing of the `BLACKDOTS` program by clicking on the `-DISPLAY` button and running the associated meta command.

Clay Session #2: Defining, Running, Recording, Displaying the Scattered Black Dots Program

```
Meta> -CLEAN
Meta> -RESET
Clay> BLACKDOTS >> BLACK 90DOT
Clay> BLACKDOTS
Meta> -RECORD
filename=.../DotsWorld/software/CIRCLEIMAGES/BLACKDOTS.jpg
Meta> -DISPLAY
BLACKDOTS >> BLACK 90DOT
DOT >> MOVE PAINT
```

Figure #2: Scattered Black Dots





```
BLACKDOTS >> BLACK 90DOT
DOT >> MOVE PAINT
```

Text Segment #3: Regions

Introduce the concept. Explain the manifesting mechanism.

Exercise #1: Scattered Black Dots

Please do the following sequence of tasks, taking great care to attend to detail, expecially with respect to the naming and positioning of the file that you will be asked to save.

1. Run Gargole, click on the dot, and then enter a new region called `LOTSOFDOTS`. This is the region that you will work in as you proceed throught this chapter of the text.

2. Mimic Clay Session #1. Note that as a side effect of doing so you will have defined precisely one program, `DOT`.
3. Leave the region that you are working in by clicking on the `LEAVE` button. Then exit the system by clicking on the `EXIT` button. Doing so will make doubly sure that any code you have written in whatever region you are working in will be saved for you. The `DOT` program isn't much, but you will regularly be writing substantial amounts of code, and you want to get into the habbit of assuring that your work is properly saved.
4. Run Gargoyle, click on the dot, and then enter the extant region called `LOTSOFDOTS`. Use the program `DOT` to generate lots of dots. If it is among the missing, something is wrong! Then initialize the system by clicking on `-CLEAN` and then `-RESET`.
5. Mimic Clay Session #2! Be sure to use the same name (`BLACKDOTS`) for the file, when prompted, that I did!
6. Leave the region that you are working in by clicking on the `LEAVE` button. Then exit the system by clicking on the `EXIT` button.
7. Create a sibling folder to the `software` folder in your `Dotsworld` folder called `web` and make a mental note that this is where you will be slowly, methodically, building a website to represent the work that you do as you work your way through this text.
8. Create a child folder called `one` within the `web` folder in your `Dotsworld` folder and make a mental note that this is where you will be putting artifacts pertaining to the work that you will be doing in this first chapter..
9. Move file that you created containing the image of the 90 black dots, the one with the name `BLACKDOTS.jpg`, to the `one` folder. You will be asked to do something with it before too long.

2.2 Scattered RGB Dots

The concept of bottom up programming is introduced in this section, as is the idea of testing the programs as you write them, provided you are using a bottom up approach. Also, a dash of color is added to the mix.

Text Segment #4: Building a Program from the Bottom Up

As I continue whatever it is I am doing with the dots, I will always make sure that I am operating in the `LOTSOFDOTS` region. So, please imagine that I have found my way there, and that I happily verified the existence of the programs `DOT` and `BLACKDOTS`, and then reinitialized the system (cleaned the canvas and restablished default values for the properties of the two featured objects.

Clay Session #3 and the accompanying Figure #3 should make good sense to you if you study the session and the figure. Please do. Still, I am going to walk you through some of my thinking as I crafted the `RGBDOTS` program.

The first three lines of Clay Session #3 define and test (twice) the `GREENDOTS` program. The first three lines of Clay Session #3 define and test (twice) the `REDDOTS` program. The third three lines of Clay Session #3 define and test (twice) the `BLUEDOTS` program. Except for the previously written `DOT` program, these are the lowest level programs of the `RGBDOTS` program that is being built. Because they don't depend on any programs no yet written, they could be tested as soons as they were defined. Figure #3 (a) shows the state of the canvas at this stage of development.

The next higher level program, one level higher than the three programs just written, is called `RGB`, and was written and tested (four times) in the five lines just after the system was reinitialized in lines 10 and 11. Figure #3 (b) shows the state of the canvas after this program was developed and tested.

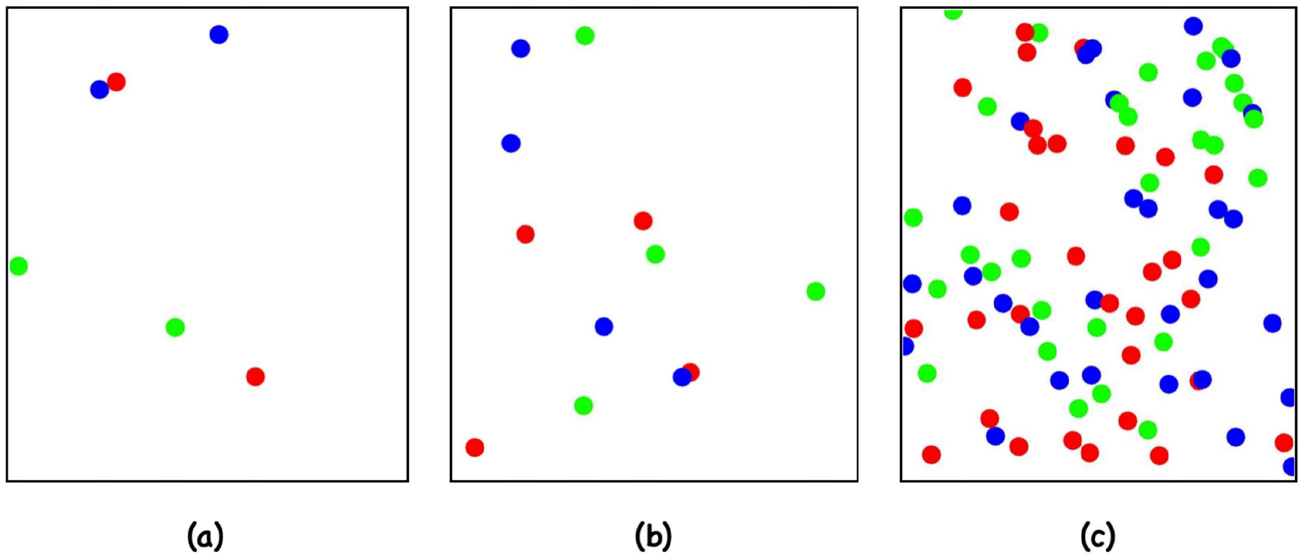
After once again reinitializing the system, the final program, the highest level program, is written and tested in the last two lines of the session. The resulting dots are shown in Figure #3 (c).

The general point to note here is that the development of this program adhered to **bottom up** programming methodology. In this methodology, one defines bits of program that rely only on previously define bits of program, and one tests as one goes.

Clay Session #3: Bottom Up Development of the RGB Dots Program

```
Clay> GREENDOT >> GREEN DOT
Clay> GREENDOT
Clay> GREENDOT
Clay> REDDOT >> RED DOT
Clay> REDDOT
Clay> REDDOT
Clay> BLUEDOT >> BLUE DOT
Clay> BLUEDOT
Clay> BLUEDOT
Meta> -CLEAN
Meta> -RESET
Clay> RGB >> REDDOT GREENDOT BLUEDOT
Clay> RGB
Clay> RGB
Clay> RGB
Clay> RGB
Meta> -CLEAN
Meta> -RESET
Clay> RGBDOTS >> 30RGB
Clay> RGBDOTS
```

Figure #3: RGB Dots



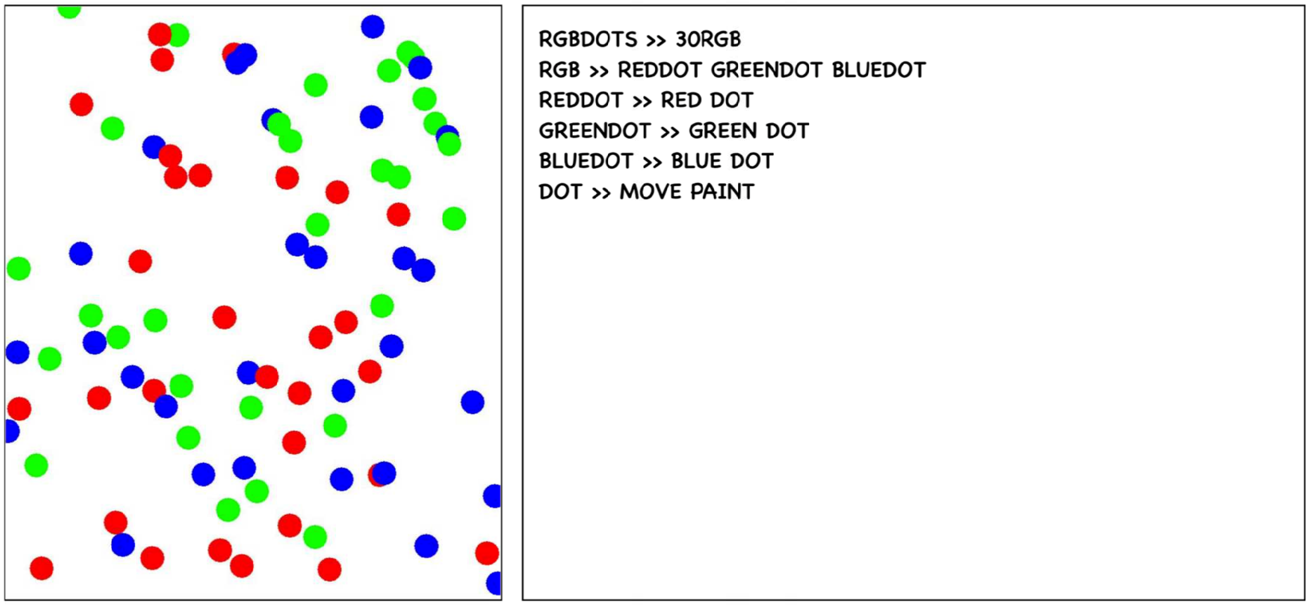
Text Segment #5: Running, Recording, and Displaying the `RGBDOTS` Program

Clay Session #4, which takes place in the `LOTSOFDOTS` region, follows the standard script for running a program, recording the image in the canvas that results from running the program, and displaying the textual program definition. Figure #4 presents the image along with a listing of the program that generated the image.

Clay Session #4: Running, Recording, Displaying the RGB Dots Program

```
Meta> -CLEAN
Meta> -RESET
Clay> RGBDOTS
Meta> -RECORD
filename=.../DotsWorld/software/CIRCLEIMAGES/RGBDOTS.jpg
Meta> -DISPLAY
RGBDOTS >> 30RGB
RGB >> REDDOT GREENDOT BLUEDOT
REDDOT >> RED DOT
GREENDOT >> GREEN DOT
BLUEDOT >> BLUE DOT
DOT >> MOVE PAINT
```

Figure #4: Scattered RGB Dots



```
RGBDOTS >> 30RGB
RGB >> REDDOT GREENDOT BLUEDOT
REDDOT >> RED DOT
GREENDOT >> GREEN DOT
BLUEDOT >> BLUE DOT
DOT >> MOVE PAINT
```

Exercise #2: Scattered RGB Dots

Please do the following sequence of tasks, taking great care to attend to detail, expecially with respect to the naming and positioning of the file that you will be asked to save.

1. Run Gargole, click on the dot, and then enter the `LOTSOFDOTS` region. Note that you will be encoding the `RGBDOTS` program using bottom up methodology.
2. Mimic Clay Session #3. Do so mindefully, to appreciate the primitive commands used, and also the process of building the `RGBDOTS` program from the bottom up.
3. Mimic Clay Session #4. Be sure to use the same name (`RBGDOTS`) for the file, when prompted, that I did!
4. Leave the region that you are working in by clicking on the `LEAVE` button. Then exit the system by clicking on the `EXIT` button.

5. Copy the file that you created containing the image of the 90 RBG dots, the file called `RGBDOTS.jpg`, to the `one` folder within the `web` folder. You will be asked to do something with it before too long.

2.3 Scattered Gray Dots

The concept of top down programming is introduced in this section. A program to paint dots in shades of gray, analogous to that written to paint red/green/blue dots, is written. Because the analogy is so direct, it is quite clear just how to proceed. Whenever you have a good idea of what you want to do, it is probably a good idea to consider using top down programming methodology.

Text Segment #6: Building a Program from the Top Down

I have found my way once again to the `LOTSOFDOTS` region. In Clay Session #5 you can see that I have listed the `RGBDOTS` program so that the details are at hand. Then, I worked by direct **analogy** with the program to write a program to paint dots in three shades of gray.

Note that I did this in a top down fashion. First I coded the highest level program (`SHADESOFGRAYDOTS`). Then I coded the midlevel program (`SHADESOFGRAYTRIO`). Finally, I coded the lowest level (other than the `DOT` program) programs to paint the three sorts of gray dots, light and medium and dark.

Coding isn't always this explicit! Sometimes, but not always. Even when it is not this direct, you might have a really good idea about how to proceed with the programming task, even without having thought through all of the details. In cases like that, the **principle of stepwise refinement** suggests that top down programming is, genarlly speaking, a good approach.

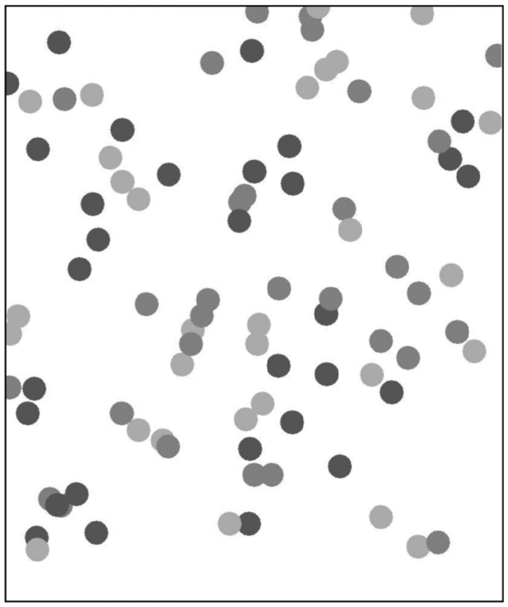
After writing the program, I ran it. It worked! If it didn't, I would have to engage in a **debugging** phase, in which case I would do something like investigate which parts of the program are working from the lowest level to the hightest level.

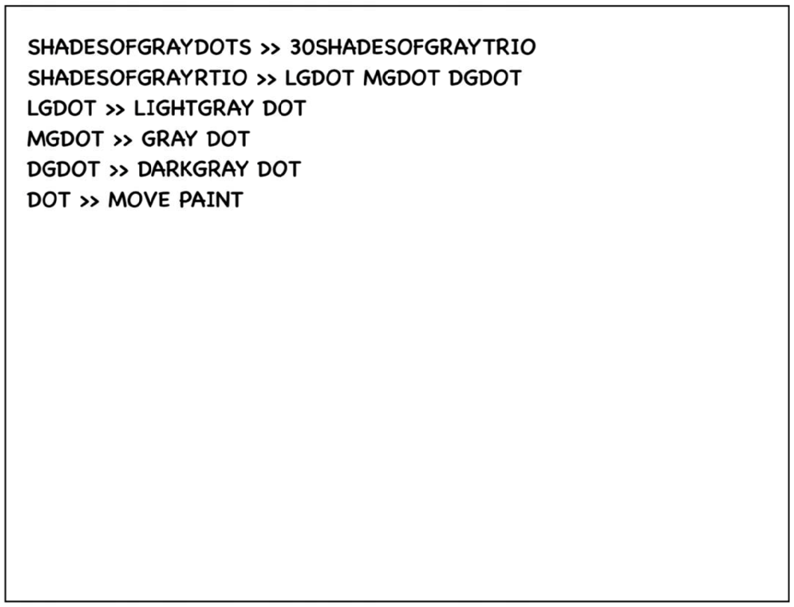
Do you see how the structure of the two programs is just the same? All I did was to replace RGB related code with "shades of gray" related code.

Clay Session #5: Top Down Development of the Shades of Gray Dots Program

```
Meta> -DISPLAY
RGBDOTS >> 30RGB
RGB >> REDDOT GREENDOT BLUEDOT
REDDOT >> RED DOT
GREENDOT >> GREEN DOT
BLUEDOT >> BLUE DOT
DOT >> MOVE PAINT
Clay> SHADESOFGRAYDOTS >> 30SHADESOFGRAYTRIO
Clay> SHADESOFGRAYRTIO >> LIGHTGRAYDOT MEDIUMGRAYDOT DARKGRAYDOT
Clay> LIGHTGRAYDOT >> LIGHTGRAY DOT
Clay> MEDIUMGRAYDOT >> GRAY DOT
Clay> DARKGRAYDOT >> DARKGRAY DOT
Clay> SHADESOFGRAYDOTS
```

Figure #5: Shades of Gray Dots





```
SHADESOFGRAYDOTS >> 30SHADESOFGRAYTRIO
SHADESOFGRAYRTIO >> LGDOT MGDOT DGDOT
LGDOT >> LIGHTGRAY DOT
MGDOT >> GRAY DOT
DGDOT >> DARKGRAY DOT
DOT >> MOVE PAINT
```

Text Segment #7: Running, Recording, and Displaying `SHADESOFGRAYDOTS`

Time to mark the fact that we did this bit of programming by reveling in the success and recording the image. To do this, we will work by direct analogy with Clay Session #4 in which we did the same thing for the `RGBDOTS` program. This is reflected in Clay Session #6.

Note that I did a bit of extra "work" by reinitializing the canvas, the circle, and the painter prior to doing the run, the record, and the display. I sometimes like doing a bit of extra work, as it helps me to form a nice habit. In this case, the nice habit is (1) setting up for an execution, (2) running the program, (3) recording the image, (4) displaying the program, and (5) reflecting once more on it. Think of this sequence of five actions as experiencing the afterglow of writing a little Clay program. It is a good habit to get into.

Clay Session #6: Running, Recording, and Displaying `SHADESOFGRAYDOTS`

```
Meta> -CLEAN
Meta> -RESET
Clay> SHADESOFGRAYDOTS
Meta> -RECORD
filename=.../DotsWorld/software/CIRCLEIMAGES/SHADESOFGRAYDOTS.jpg
Meta> -DISPLAY
SHADESOFGRAYDOTS >> 30SHADESOFGRAYTRIO
SHADESOFGRAYRTIO >> LIGHTGRAYDOT MEDIUMGRAYDOT DARKGRAYDOT
LIGHTGRAYDOT >> LIGHTGRAY DOT
MEDIUMGRAYDOT >> GRAY DOT
DARKGRAYDOT >> DARKGRAY DOT
DOT >> MOVE PAINT
```

Exercise #3: Scattered Shades of Gray Dots

Please do the following sequence of tasks, taking great care to attend to detail, expecially with respect to the naming and positioning of the file that you will be asked to save.

1. Run Gargole, click on the dot, and then enter a new region called `LOTSOFDOTS`. Note that you will be encoding the `SHADESOFGRAYDOTS` program using top down methodology.
2. Mimic Clay Session #5!
3. Mimic Clay Session #6! Be sure to use the same name (`SHADESOFGRAYDOTS`) for the file, when prompted, that I did!
4. Leave the region that you are working in by clicking on the `LEAVE` button. Then exit the system by clicking on the `EXIT` button.
5. Copy the file that you created containing the image of the 90 gray (30 light, 30 medium, 30 dark) dots, the file called `SHADESOFGRAYDOTS.jpg`, to the `one` folder within the `web` folder. You will be asked to do something with it before too long.

2.4 Scattered Colored Dots

This section features a program to generate randomly colored dots. It is a very simple program! Generally speaking, the more relaxed the constraints, the easier it is to write a program.

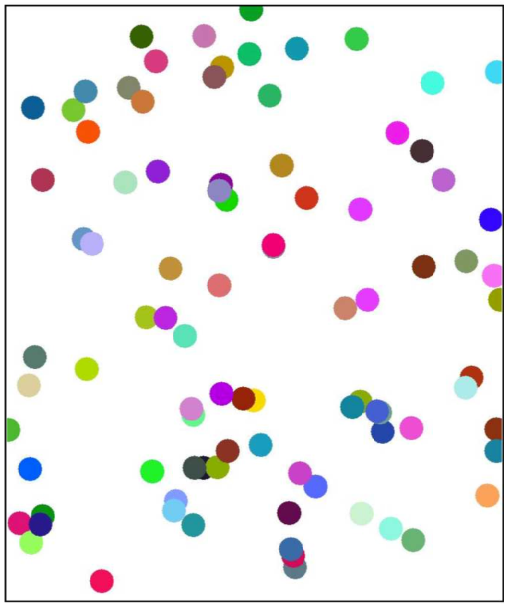
Text Segment #8: Just Doing It

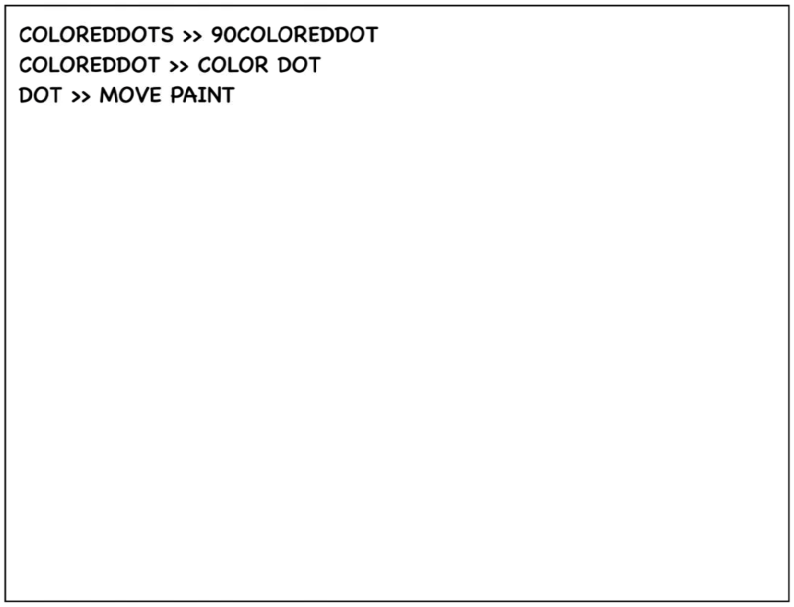
I am in the `LOTSOFDOTS` region. In Clay Session #7 you can see the program. There is not much to be said, beyond noting that the `color` primitive binds the painter's metaphoric brush to a random color.

Clay Session #7: Randomly Colored Dots Program

```
Clay> COLOREDDOTS >> 90COLOREDDOT
Clay> COLOREDDOT >> COLOR DOT
Clay> DOT >> MOVE PAINT
Clay> COLOREDDOTS
Meta> -RECORD
filename=.../DotsWorld/software/CIRCLEIMAGES/COLOREDDOTS.jpg
Meta> -DISPLAY
COLOREDDOTS >> 90COLOREDDOT
COLOREDDOT >> COLOR DOT
DOT >> MOVE PAINT
```

Figure #6: Randomly Colored Dots





```
COLOREDDOTS >> 90COLOREDDOT
COLOREDDOT >> COLOR DOT
DOT >> MOVE PAINT
```

Exercise #4: Scattered Randomly Colored Dots

Please do the following sequence of tasks, taking great care to attend to detail, expecially with respect to the naming and positioning of the file that you will be asked to save.

1. Run Gargole, click on the dot, and then enter the `LOTSOFDOTS` region.
2. Mimic Clay Session #7! Be sure to use the same name (`COLOREDDOTS`) for the file, when prompted, that I did!
3. Leave the region that you are working in by clicking on the `LEAVE` button. Then exit the system by clicking on the `EXIT` button.
4. Copy the file that you created containing the image of the 90 randomly colored dots, the file called `COLOREDDOTS.jpg`, to the `one` folder within the `web` folder. You will be asked to do something with it before too long.

2.5 Starry Night

The concept of **emergence** is featured in the program to be written in this section. The concept of emergence will be introduced abstractly, discussed briefly, and then illustrated as the program is executed.

Text Segment #9: Complex Systems and Emergence

A **complex system** is a system composed of a number of relatively simple objects that interact in well defined ways to produce an unanticipated effect. The unanticipated effect is known as an **emergent phenomenon**. For example, the composer Phillip Glass combines simple sequences of notes *of equal duration*in repetitive ways that produce mesmorizing rhythms. The rhythms can be seen as an emergent phenomena. On a much grander level, theorists suggest that consiousness is an emergent phenomenon with respect to the neuronal interactions in the brain.

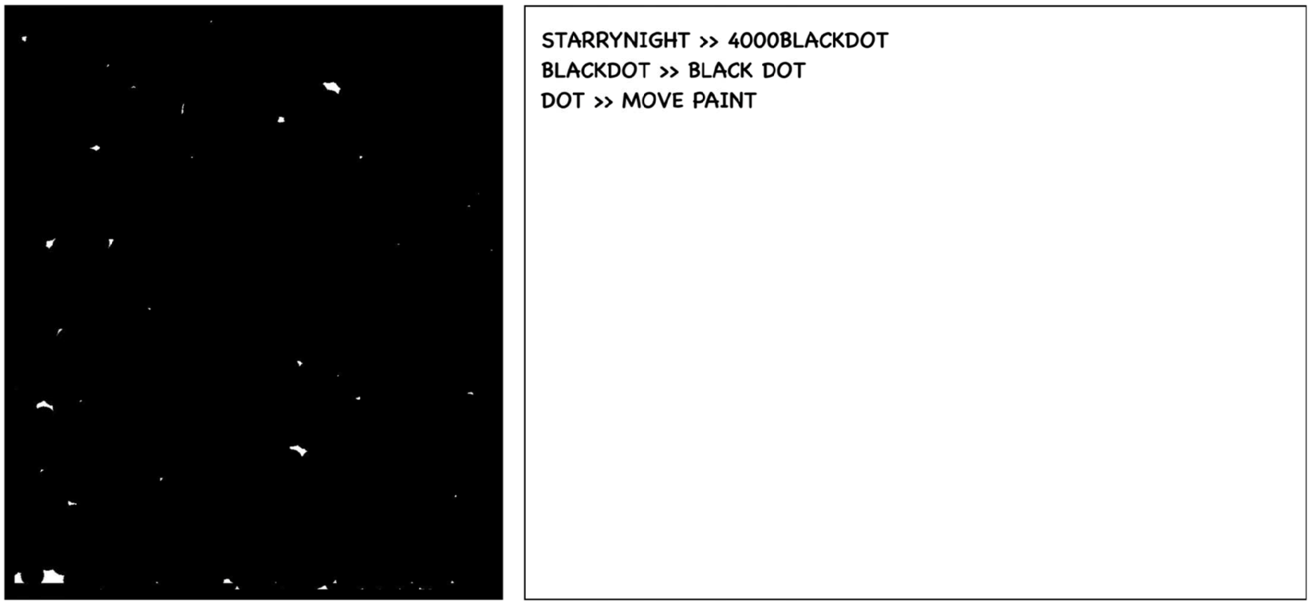
In Clay Session #8 you see a program that repeatedly paints black dots in random areas of the screen. Eventually, one stops seening black dots and one starts seeing points of light – which are intended to represent stars in the night sky. The stars emerge as the dots "paint the daytime black."

How did I come up with 4000? I just tried a few numbers. This is the one I liked best. On my machine it didn't take too long to run, and the effect was what I had in mind.

Clay Session #8: Starry Night

```
STARRYNIGHT >> 4000BLACKDOT
BLACKDOT >> BLACK DOT
DOT >> MOVE PAINT
```

Figure #7: Starry Night



```
STARRYNIGHT >> 4000BLACKDOT
BLACKDOT >> BLACK DOT
DOT >> MOVE PAINT
```

Exercise #5: Starry Night

Please do the following sequence of tasks, taking great care to attend to detail, expecially with respect to the naming and positioning of the file that you will be asked to save.

1. Run Gargole, click on the dot, and then enter the `LOTSOFDOTS` region.
2. Mimic Clay Session #8! Be sure to use the same name (`STARRYNIGHT`) for the file, when prompted, that I did!
3. Leave the region that you are working in by clicking on the `LEAVE` button. Then exit the system by clicking on the `EXIT` button.
4. Copy the file that you created containing the starry night image, the file called `STARRYNIGHT.jpg`, to the `one` folder within the `web` folder. You will be asked to do something with it before too long.

2.6 The Rest of the Exercises

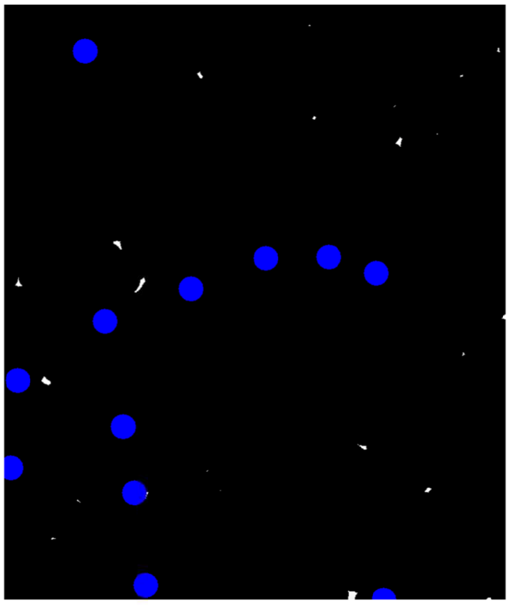
Up until this point your work has involved reading, following along, and mimicking some programming. Now you will be invited to write some programs on your own.

Exercise #6: Nightsky

Please do the following sequence of tasks, taking great care to attend to detail, expecially with respect to the naming and positioning of the file that you will be asked to save.

1. Run Gargole, click on the dot, and then enter the `LOTSOFDOTS` region.
2. Write a program called `NIGHTSKY` which will render 12 blue moons in a starry sky. An example run of the program that I wrote appears in Figure #8.
3. Do the 5 step afterglow thing, taking care to call your file `NIGHTSKY` when prompted.
4. Leave the region that you are working in by clicking on the `LEAVE` button. Then exit the system by clicking on the `EXIT` button.
5. Copy the file that you created containing the night sky image (emergent stars and explicit blue moons), the file called `NIGHTSKY.jpg`, to the `one` folder within the `web` folder. You will be asked to do something with it before too long.

Figure #8: Night Sky



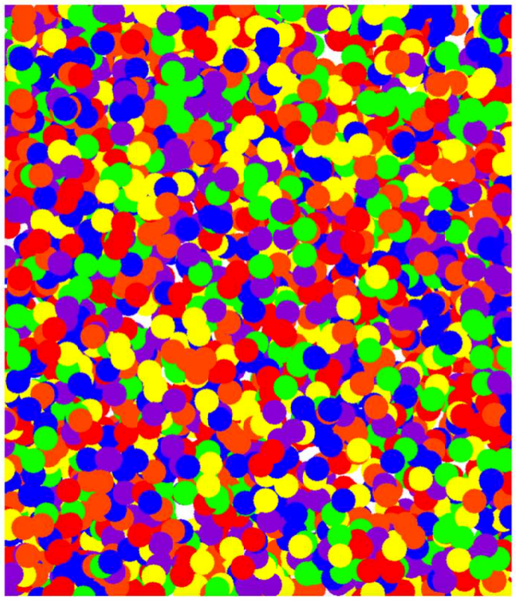
Break the problem of generating the image into two parts. First, paint the starry sky. Second, paint the 12 blue moons. Correspondingly, the top level program should be written in terms of exactly two commands.

Exercise #7: Rainbow of Dots

Please do the following sequence of tasks, taking great care to attend to detail, expecially with respect to the naming and positioning of the file that you will be asked to save.

1. Run Gargole, click on the dot, and then enter the `LOTSOFDOTS` region.
2. Write a program called `RAINBOWOFDOTS` which will render 3000 dots, 500 each of colors red, blue, yellow, green, orange, and violet (sorry, indigo!), taking care that no dot color regularly tramples on any other dot color. An example run of the program that I wrote appears in Figure #8 (b).
3. Do the 5 step afterglow thing, taking care to call your file `RAIMBOWOFDOTS` when prompted.
4. Leave the region that you are working in by clicking on the `LEAVE` button. Then exit the system by clicking on the `EXIT` button.
5. Copy the file that you created containing the 3000 dots equally distributed over the specified six color palette, the file called `RAINBOWOFDOTS.jpg`, to the `one` folder within the `web` folder. You will be asked to do something with it before too long.

Figure #9: Rainbow of Dots



Please remember that there are only six colors of dots. Furthermore, please remember that no colored dot consistently tramples on dots of any other color.

Exercise #8: Weighted Unordered Green/Orange/Violet (GOV) Dots

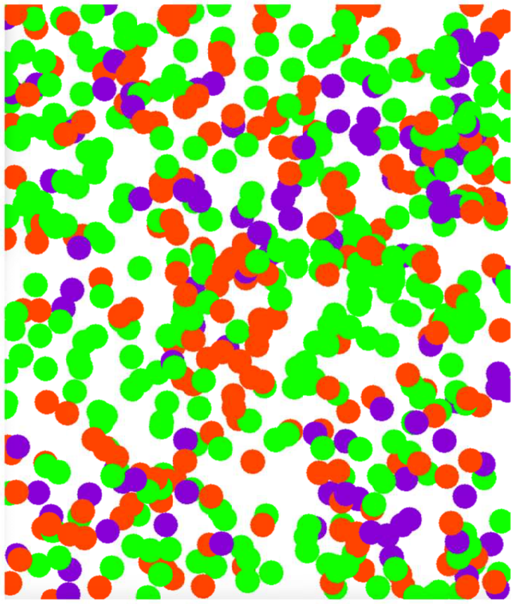
Please do the following sequence of tasks, taking great care to attend to detail, expecially with respect to the naming and positioning of the file that you will be asked to save.

1. Run Gargole, click on the dot, and then enter the `LOTSOFDOTS` region.
2. Write a program called `GOVDOTSWU` which will render 600 dots such there are twice as many orange dots as violet dots and three times as many green dots as violet dots, and such that no dot of any one color consistently tramples on the dots of any other color. (For example, sometimes a green dot will cover an orange dot and

sometimes an orange dot will cover a green dot.) An example run of the program that I wrote appears in Figure #8 (c).

3. Do the 5 step afterglow thing, taking care to call your file `GOVDOTSWU` when prompted.
4. Leave the region that you are working in by clicking on the `LEAVE` button. Then exit the system by clicking on the `EXIT` button.
5. Copy the file that you created containing the 600 dots according to the specified constraints, the file called `GOVDOTSWU.jpg`, to the `one` folder within the `web` folder. You will be asked to do something with it before too long.

Figure #10: Green / Orange / Violet Dots, Weighted and Unordered



Please don't forget that there are there are 300 grren dots, 200 orange dots, and 100 violet dots. Furthermore, please keep in mind that no color regularly tramples on any other color.

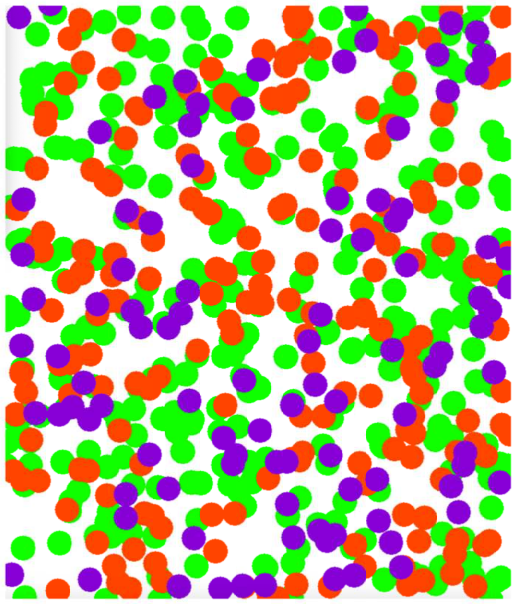
Exercise #9: Weighted Ordered Green/Orange/Violet (GOV) Dots

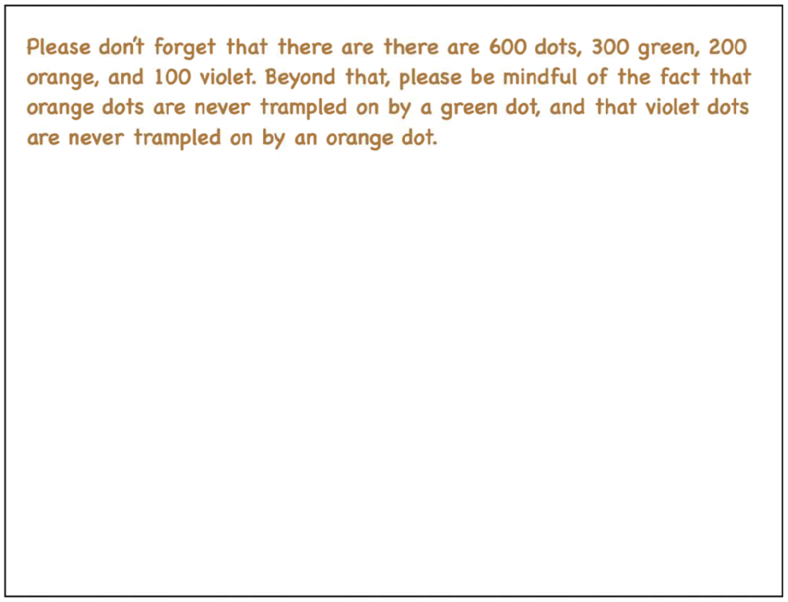
Please do the following sequence of tasks, taking great care to attend to detail, expecially with respect to the naming and positioning of the file that you will be asked to save.

1. Run Gargole, click on the dot, and then enter the `LOTSOFDOTS` region.
2. Write a program called `GOVDOTSWO` which will render 600 dots such there are twice as many orange dots as violet dots and three times as many green dots as violet dots, but such that no green dot ever covers an orange dot, and no orange dot ever covers a violet dot. An example run of the program that I wrote appears in Figure #8 (d).
3. Do the 5 step afterglow thing, taking care to call your file `GOVDOTSWO` when prompted.
4. Leave the region that you are working in by clicking on the `LEAVE` button. Then exit the system by clicking on the `EXIT` button.
5. Copy the file that you created containing the 600 dots according to the specified constraints, the file called `GOVDOTSWO.jpg`, to the `one` folder within the `web` folder. You will be asked to do something with it before too

long.

Figure #11: Green / Orange / Violet Dots, Weighted and Ordered





Please don't forget that there are there are 600 dots, 300 green, 200 orange, and 100 violet. Beyond that, please be mindful of the fact that orange dots are never trampled on by a green dot, and that violet dots are never trampled on by an orange dot.

Exercise #10: Web Page

This exercise pertains to establishing a web page though which to view your work with respect the this chapter. You have already done most of the work on this page, in that you have developed most of its content by generating and saving images resulting from Clay programs. Moreover, I have done some of the work for you by writing a bit of html code which references the files that you have created (provided you were careful to name them in the prescribed way). You have just a bit of work to do which takes the form of downloading a couple of fiels and generating a text file consisting the the programs that you wrote as a result of working through this chapter.

1. Find your way to the web page for this text and check out the example web page for this chapter that I made available from the page. After you finish this exercise, you will have created a very similar sort of web page.
2. From the course web page, please download the file `LotsOfDots.html` to the folder called `one` where you have been storing the image files generated in the Dotsworld.
3. From this same page, download the file `style.css` to your `website` folder.
4. Load the `LotsOfDots.html` file into a Web browser. Perhaps you can do this merely by double clicking on the file.
5. Check to see that the images are found when you click on the referencing links. If not, something is wrong, and you should look into fixing it.
6. Next, create a text file containing lisings of the programs featured in this chapter, each sepated by a line of white space I will look quite like the file that I posted on the sample page that I asked you to consider. To do so:
   (a) Run the Gargoyle program, click on the dot, and enter the `LOTSOFDOTS` region.
   (b) Set the history flag to on, by clicking on the `HISTORY` button and seleting `ON`.

(c) Using the `-DISPLAY` button nine times, display each of the following programs to the Clay output window: BLACKDOTS, RGBDOTS, SHADESOFGRAYDOTS, COLOREDDOTS, STARRYNIGHT, NIGHTSKY, RAINBOWOFDOTS, GOVDOTSWU, and GOVDOTSWO.

(d) Select all of the text in the Clay output window, copy it, and paste it into a buffer in your favorite text editor. Replace the nine lines beginning with the word `META` with an empty line. And finally save the buffer as a file called `PROGRAMS.text` the the folder called `one` of the `website` folder.

7. Reload the `LotsOfDots.html` file in your browser, and see if you can reference the file that you just created containing the listings of the nine programs. If you can, yay! If not, please look into what went wrong.

Exercise #11: The Powerful Ideas

Do your best, based mainly on the text, but incorporating relevant information that you find via Google search, if you like, write a one paragraph description of each of the following powerful ideas. Do it by hand, in the space provided, if you are working from a hard copy of this text. Otherwise, do it in whatever reasonable place you like.

Bottom Up Programming

Top Down Programming

Coding by Analogy

Debugging (debugging the mind by debugging the machine)

Emergence in Complex Systems

2.7 Clay Recap

This section reviews the Clay commands and the Meta commands that were used in this chapter, and it also summarizes the Clay constructs that were introduced.

Primitive Clay Commands and Constructs

The following list of commands is a complete list of the Clay primitive commands that were used in this chapter. It is noteworthy that there are so few, particularly in light of the fact that most of them pertain to changing the color on the painters brush!

- BLACK
- MOVE
- PAINT
- RED
- GREEN
- BLUE
- COLOR
- LIGHTGRAY
- GRAY
- DARKGRAY
- YELLOW
- ORANGE
- VIOLET

Clay Constructs

Clay was designed to be a very simple symbolic language. It lacks most of the constructs of convenience that one typically finds in programming languages. The point of the language is largely to see how much you can do with very little. With Clay, the question isn't generally "What can it do?", but "What can I do with it?", affording you opportunities to hone your thinking with respect to fundamental computation.

- Sequence: You can write down a sequence of Clay commands.
- Repetition: You can prepend a positive number to a Clay command and the command will be executed the indicated number of times.
- Definition: You can name a sequence of commands by writing an identifier (command name) followed by the definition symbol (`>>`) followed by the sequence of commands.

Meta Commands

The following list of meta commands, commands designed to perform infrastructural functions in support of programming, is a complete list of the meta commands that were introduced in this chapter.

- -ENTER :: Enter a new subregion of the current region in which do do some work, creating the region of it does not yet exist.
- -LEAVE – Assuming that you are not in the top level region, leave the current subregion of a region and save the programs for subesquent use.
- -EXIT – Terminate execution of the Gargoyle system, saving programs in the current region and its ancestor regions prior to doing so.
- -RESET :: Bind all properties of the featured objects to their default values.
- -CLEAN :: Erase the contents of the painter's canvas.
- -RECORD :: Save the contents of the painter's canvas as a `.jpg` file.
- -DISPLAY :: List a Clay program in the Clay output window, in a manner that is ordered topologically in a top down fashion.
- -HISTORY :: Set or reset a flag (you are prompted to select the option) which determines whether or not the Clay output window will be cleared immediately after input has been recieved from the Clay input box.

3 Rings and Things

? on ?

3.1 Expanding and Shrinking

There are commands for expanding the circle and shrinking the circle in specific ways that are dependent on the size of the circle. There are also commands for increasing or decreasing the size of the circle that are independent of the size of the circle. We will introduce the commands and we will make use of them to solve some simple problems in ways that exhibit powerful ideas.

Text Segment #1: Expanding, Shrinking, Drawing and Invariance

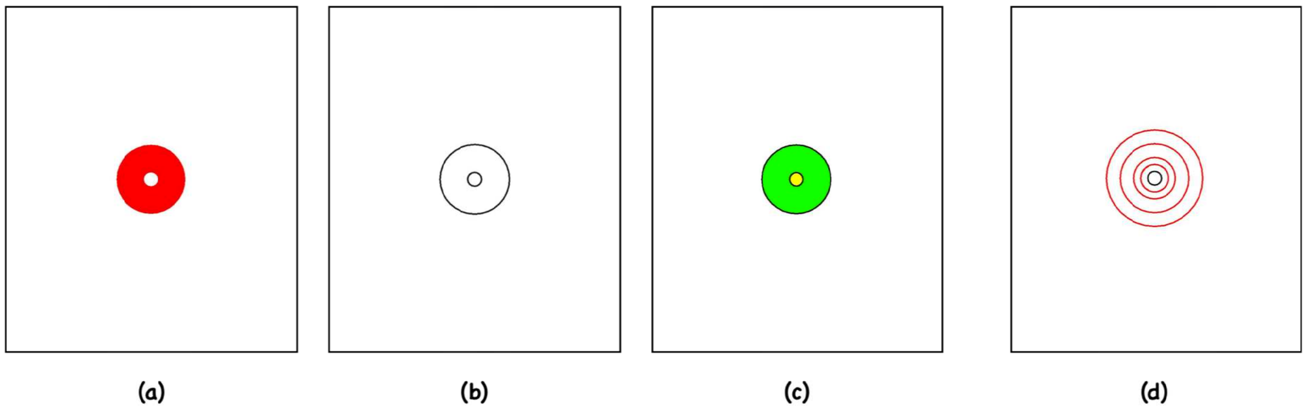
There are commands to expand the circle by a factor of two or three or five or seven. These commands are named: `x2` and `x3` and `x5` and `x7`. Likewise, there are commands to shrink the circle by a factor of two or three or five or seven. These commands are named: `s2` and `s3` and `s5` and `s7`. Figure #1 illustrates the use of these commands. Line 1 generates the image in Figure #1 (a). Line 4 generates the image in Figure #1 (b). Line 7 generates the image in Figure #1 (c). Lines 10-13 define four commands that make an effor to be *invariant with respect to size*.

A command is **invariant with respect to a property** if the value of the property is the same after the command has been executed as it was before execution of the command began. This is a powerful idea that underlies the idea of *reuse* in computer programming, and consequently is essential to the architecting of sound computer programs. For a real world example, suppose you are in need of entering a room through a closed door. If, after opening the door and passing through, you are good enough to close the door, your passage into the room was invariant with respect to the door. It may be worth mentioning that the `PRIMECIRCLES` is also invariant with respect to circle size, but this is due to the fact that all of its subcommands are invariant with respect to circle size. The last line of the session generates the image shown in Figure #1 (d).

Clay Session #1: Expanding and Shrinking and Drawing

```
Clay> RED X5 PAINT S5 WHITE PAINT
Meta> -RESET
Meta> -CLEAN
Clay> X5 DRAW S5 DRAW
Meta> -RESET
Meta> -CLEAN
Clay> X5 GREEN PAINT BLACK DRAW S5 YELLOW PAINT BLACK DRAW
Meta> -RESET
Meta> -CLEAN
Clay> DRAW2 >> X2 DRAW S2
Clay> DRAW3 >> X3 DRAW S3
Clay> DRAW5 >> X5 DRAW S5
Clay> DRAW7 >> X7 DRAW S7
Clay> PRIMECIRCLES >> DRAW2 DRAW3 DRAW5 DRAW7
Clay> DRAW
Clay> RED PRIMECIRCLES
```

Figure #1: Expanding and Shrinking and Drawing



Text Segment #2: The Fundamental Theorem of Arithmetic to the Rescue!

What is with restricting expansion and shrinkage to prime factors? The answer is that they serve as just the right set of "building blocks" for defining commands to expand and shrink the circle by arbitrary factors. This is simply because, according to the fundamenatl theorem of arithmetic, any positive integer can be reached by an appropriate multiplication of primes. This isn't the place for the lovely proof. Rather, a program will be presented that makes use of the idea. The first five lines of Clay Session #2 extends the set of invariant circle drawing commands defined in Clay Session #1. Lines 6 and 7 then define and use a command, called `CIRCLES` to draw 10 circles with a linear progression of radii, regardless of what the circle size happens to be prior to running the `CIRCLES` command. The image generated by running the command is shown in Figure #2 (a).

The image shown in Figure #2 (b) consists of those same ten circles, but with the odd circles drawn in red and the even circles drawn in blue. The program written in lines 10-12 of Clay Session #2 generates the image when it is run in line 13. The noteworthy aspect of this program is that it explicitly makes use of the problem solving strategy of *problem decomposision*, another powerful idea. **Problem decomposition** is the problem solving strategy of decomposing a problem into subproblems, solving the subproblems, and then composing a solution to the original problem by means of the solutions to the subproblems. In this case, the problem of drawing the red and blue circles was decomposed into the problem of drawing the red circles and the problem of drawing the blue circles.

Clay Session #2: Ten Concentric Circles

```
Clay> DRAW4 >> 2X2 DRAW 2S2
Clay> DRAW6 >> X3 X2 DRAW S2 S3
Clay> DRAW8 >> 3X2 DRAW 3S2
Clay> DRAW9 >> 2X3 DRAW 2S3
Clay> DRAW10 >> X5 X2 DRAW S2 S5
Clay> CIRCLES >> DRAW1 DRAW2 DRAW3 DRAW4 DRAW5 DRAW6 DRAW7 DRAW8 DRAW9 DRAW10
Clay> DRAW1 >> DRAW
Clay> CIRCLES
META> -RECORD
filename=.../DotsWorld/software/CIRCLEIMAGES/CIRCLES.jpg
META> -DISPLAY
CIRCLES >> DRAW1 DRAW2 DRAW3 DRAW4 DRAW5 DRAW6 DRAW7 DRAW8 DRAW9 DRAW10
Clay> DRAW1 >> DRAW
Clay> DRAW2 >> X2 DRAW S2
Clay> DRAW3 >> X3 DRAW S3
Clay> DRAW4 >> 2X2 DRAW 2S2
Clay> DRAW5 >> X5 DRAW S5
Clay> DRAW6 >> X3 X2 DRAW S2 S3
Clay> DRAW7 >> X7 DRAW S7
Clay> DRAW8 >> 3X2 DRAW 3S2
Clay> DRAW9 >> 2X3 DRAW 2S3
Clay> DRAW10 >> X5 X2 DRAW S2 S5
```

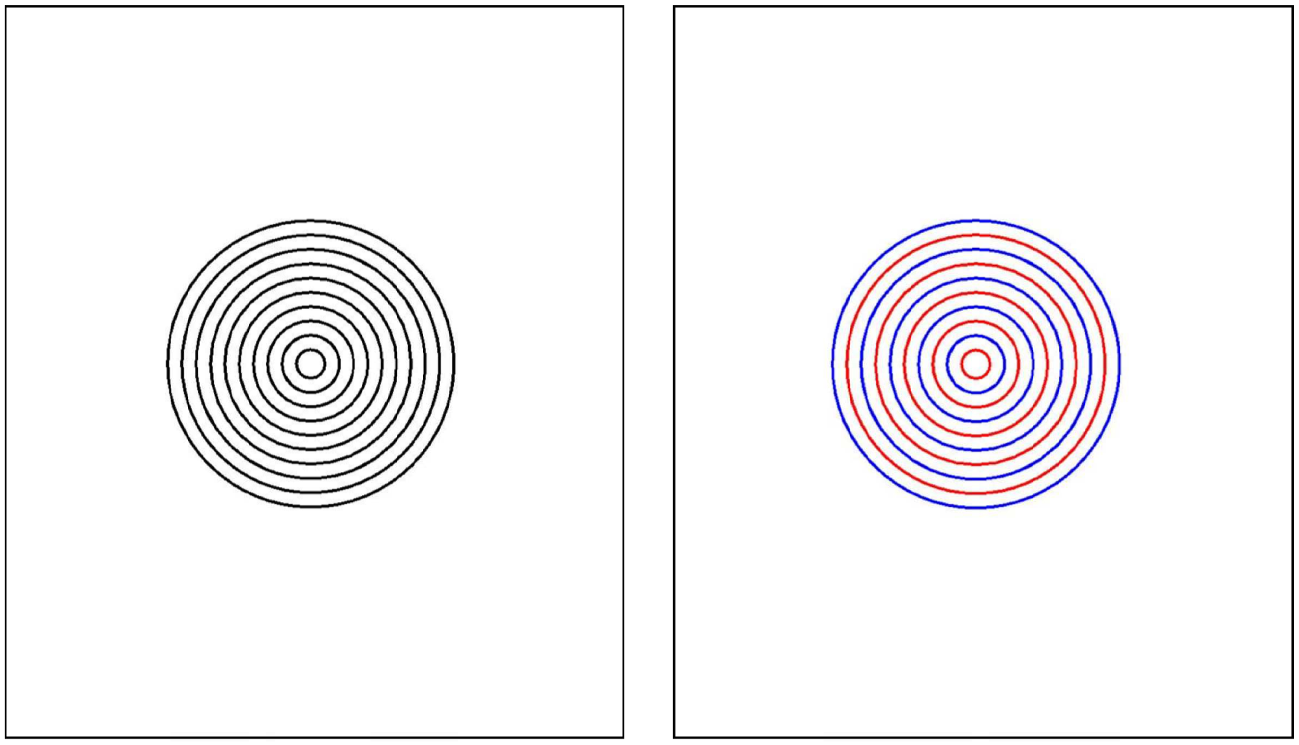
Text Segment #3: Problem Decomposition

The image shown in Figure #2 (b) consists of those same ten circles, but with the odd circles drawn in red and the even circles drawn in blue. The program written in lines 1-3 of Clay Session #3 generates the image when it is run in line 4. The noteworthy aspect of this program is that it explicitly makes use of the problem solving strategy of *problem decomposision*, another powerful idea. **Problem decomposition** is the problem solving strategy of decomposing a problem into subproblems, solving the subproblems, and then composing a solution to the original problem by means of the solutions to the subproblems. In this case, the problem of drawing the red and blue circles was decomposed into the problem of drawing the red circles and the problem of drawing the blue circles.

Clay Session #3: Ten Concentric Circles Partitioned by Color

```
Clay> BICOLORCIRCLES >> REDCIRCLES BLUECIRCLES
Clay> REDCIRCLES >> RED DRAW1 DRAW3 DRAW5 DRAW7 DRAW9
Clay> BLUECIRCLES >> BLUE DRAW2 DRAW4 DRAW6 DRAW8 DRAW10
Clay> BICOLORCIRCLES
META> -RECORD
filename=.../DotsWorld/software/CIRCLEIMAGES/BICOLORCIRCLES.jpg
META> -DISPLAY
BICOLORCIRCLES >> REDCIRCLES BLUECIRCLES
REDCIRCLES >> RED DRAW1 DRAW3 DRAW5 DRAW7 DRAW9
BLUECIRCLES >> BLUE DRAW2 DRAW4 DRAW6 DRAW8 DRAW10
```

Figure #2: Ten Circles, and the Ten Circles Partitioned by Color


(a) (b)

Exercise #1: Ten Concentric Circles Partitioned by Color

1. Create a region called `RINGSANDTHINGS` within your top level region. Then enter your newly created region.
2. Within your `RINGSANDTHINGS` region, define the `BICOLORCIRCLES`.
3. Do the afterglow thing, being sure to type the name `BICOLORCIRCLES` when prompted by the `-RECORD` meta command.
4. Create a new folder called `two` within your `website` folder as a sibling to the `one` folder.
5. Move the file called `BICOLORCIRCLES.jpg` into the folder called `two` of your `website` folder.

3.2 Permutation Rings

An modest application of the resizing commands will now be presented, largely as a vehicle for suggesting the importance of attending to detail in programming activities.

Text Segment #4: Permutations and Permutation Sets

The permutation set on a set of elements is the collection of all distince sequences of the elements in the set. For example, the permutation set of {A,C,T} is { <A,C,T>, <A,T,C>, <C,A,T>, <C,T,A>, <T,A,C>, <T,C,A>, }. The permutation set of the single digit primes contains 24 elements. Can you write it out? Any idea how many permutations in the set of all permutations of {Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday}?

Text Segment #5: Permutation Ring Thing Program

The program featured in Clay Session #4 generates and displays, loosely speaking, the set of permutation ring things over the three colors. I took pains to develop the program very methodically in a strict bottom up manner. Please note the methodical testing, and even the methodical introduction of the names for the permutation rings (alphabetical by first letter of color). Successful programmers tend to be a bit pathological about details like this!

The final line of Clay Session #4 produced the image shown in Figure #3 (a). Unfortuanately, for me, the image shown in Figure #3 (b) only materialized after clearing the screen and running the program 37 more times. I counted! I was insistant upon obtaining an image in which the six permutation ring things neither overlapped one another nor overstepped the boundary of the canvas. Clearly, I can be patient when I want to be!

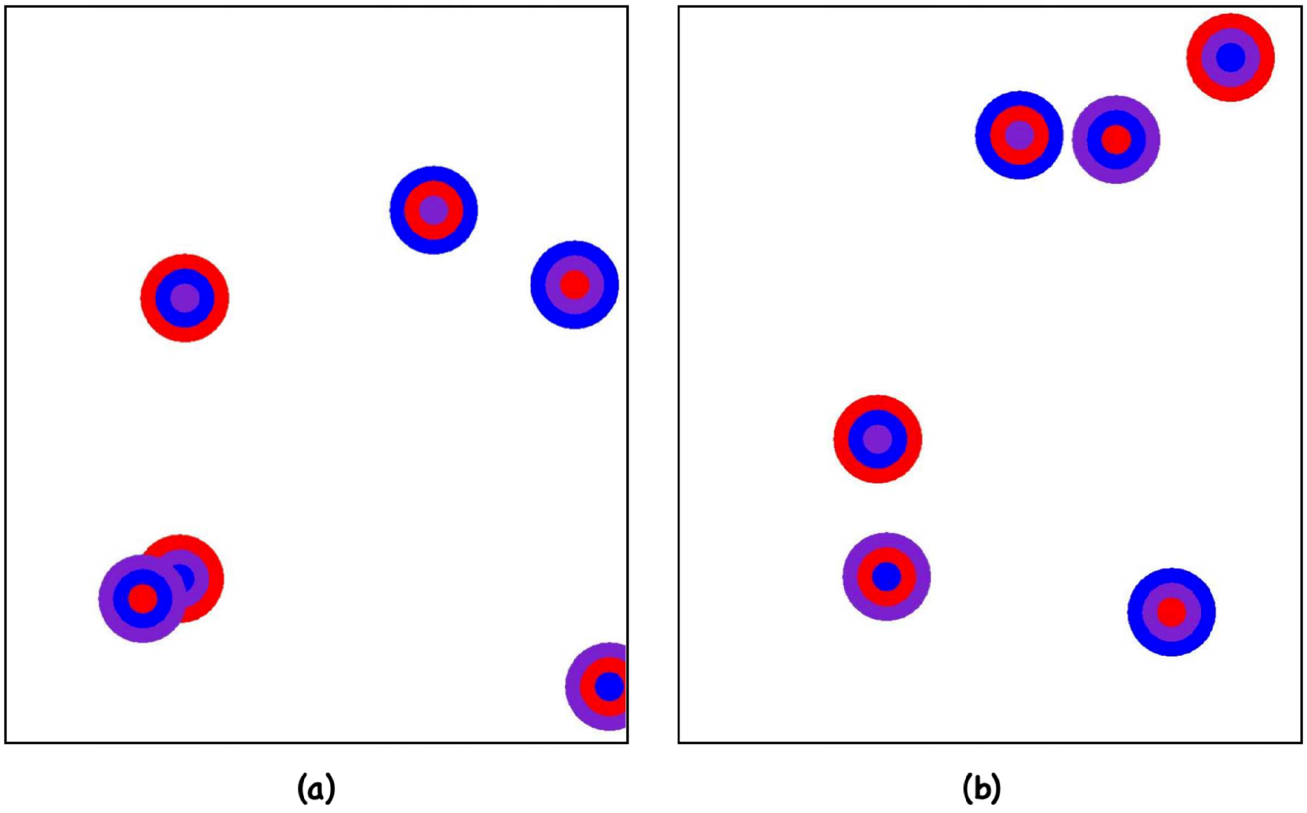
Clay Session #4: Permutation Ring Thing Program from the Bottom Up

```
Clay> BDOT >> X3 PAINT S3
Clay> VIOLET BDOT
Clay> MDOT >> X2 PAINT S2
Clay> BLUE MDOT
Clay> SDOT >> PAINT
Clay> RED SDOT
Clay> BRV >> BLUE BDOT RED MDOT VIOLET SDOT
Clay> BRV
Clay> BVR >> BLUE BDOT VIOLET MDOT RED SDOT
Clay> BVR
Clay> RBV >> RED BDOT BLUE MDOT VIOLET SDOT
Clay> RBV
Clay> RVB >> RED BDOT VIOLET MDOT BLUE SDOT
Clay> RVB
Clay> VBR >> VIOLET BDOT BLUE MDOT RED SDOT
Clay> VBR
Clay> VRB >> VIOLET BDOT RED MDOT BLUE SDOT
Clay> VRB
Meta> -CLEAN
Meta> -RESET
Clay> P1 >> MOVE BRV
Clay> P1
Clay> P2 >> MOVE BVR
Clay> P2
Clay> P3 >> MOVE RBV
Clay> P3
Clay> P4 >> MOVE RVB
Clay> P4
Clay> P5 >> MOVE VBR
Clay> P5
Clay> P6 >> MOVE VRB
Clay> P6
Meta> -CLEAN
Meta> -RESET
Clay> PERMUTATIONRINGTHINGS >> P1 P2 P3 P4 P5 P6
Clay> PERMUTATIONRINGTHINGS
```

Clay Session #5: Afterglowing the Permutation Ring Thing Program

```
Clay> PERMUTATIONRINGTHINGS
Meta> -RECORD
filename=.../DotsWorld/software/CIRCLEIMAGES/PERMUTATIONRINGTHINGS.jpg
Meta> -DISPLAY
PERMUTATIONRINGTHINGS >> P1 P2 P3 P4 P5 P6
P1 >> MOVE BRV
P2 >> MOVE BVR
P3 >> MOVE RBV
P4 >> MOVE RVB
P5 >> MOVE VBR
P6 >> MOVE VRB
BRV >> BLUE BDOT RED MDOT VIOLET SDOT
BVR >> BLUE BDOT VIOLET MDOT RED SDOT
RBV >> RED BDOT BLUE MDOT VIOLET SDOT
RVB >> RED BDOT VIOLET MDOT BLUE SDOT
VBR >> VIOLET BDOT BLUE MDOT RED SDOT
VRB >> VIOLET BDOT RED MDOT BLUE SDOT
BDOT >> X3 PAINT S3
MDOT >> X2 PAINT S2
SDOT >> PAINT
```

Figure #3: Permutation Ring Thing Images



Exercise #2: Permutation Ring Things

1. Enter the RINGSANDTHINGS region within your top level region.
2. Within your RINGSANDTHINGS region, define the PERMUTATIONRINGTHINGS.
3. Do the afterglow thing, being sure to type the name PERMUTATIONRINGTHINGS when prompted by the -RECORD meta command.
4. Move the file called PERMUTATIONRINGTHINGS.jpg into the folder called two of your website folder.

3.3 Incrementing and Decrementing

There is a command to increment the circle radius by 1 unit and a command to decrement the circle radius by 1 unit. These commands provide more flexability than the the previously introduced expand and shrink commands.

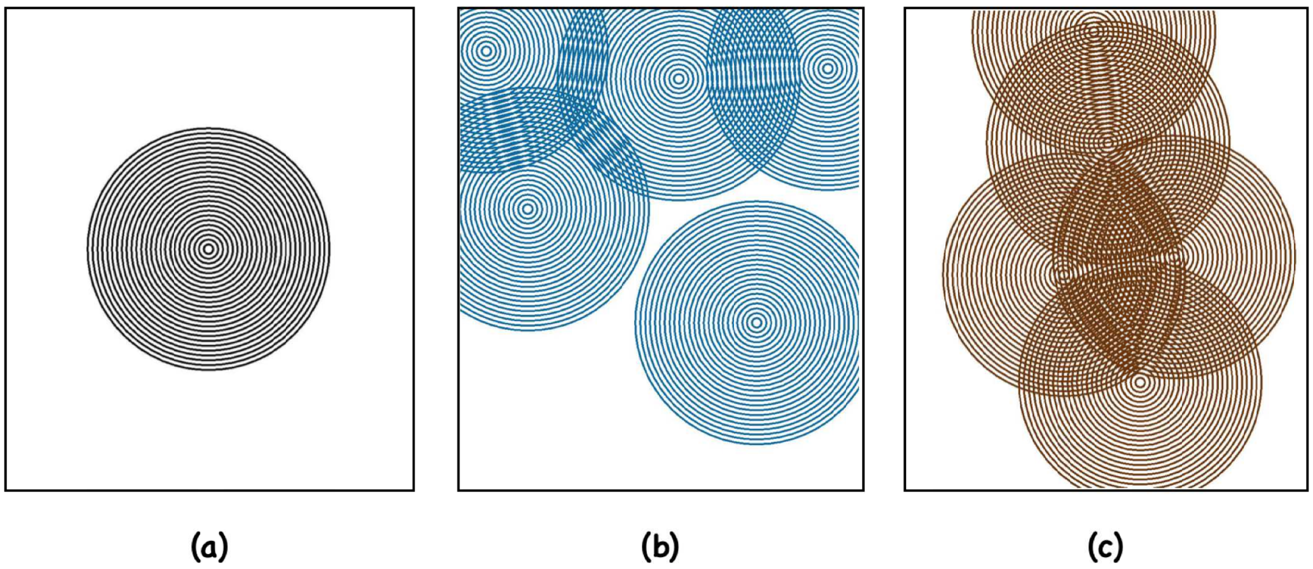
Text Segment #6: Concentric Circles

The INC command increases the circle size by 1 unit. The DEC command decreases the circle size by 1 unit. The first clump of code in Clay Session #? captures the top down programming of a command to draw 25 concentric circles of radii 10, 20, 30, ..., 250. The remainder of the session consists of running the program, recording the image, displaying the program, and casually generating a couple of additional images, one blue and one orange. Figure #? shows these three images.

Clay Session #6: Concentric Circles

```
Clay> CONCENTRICCIRCLES >> TOPOINT 25CIRCLEPLUS 250DEC FROMPOINT
Clay> TOPOINT >> 20DEC
Clay> FROMPOINT >> 20INC
Clay> CIRCLEPLUS >> 10INC DRAW

Clay> CONCENTRICCIRCLES

Meta> -RECORD
fileName = CONCENTRICCIRCLES
filename=/Users/blue/blues/2020Q/authoring/DotsWorld/software/CIRCLEIMAGES/CONCENTRICCIRCLES.jpg

Meta> -DISPLAY
CONCENTRICCIRCLES >> TOPOINT 25CIRCLEPLUS 250DEC FROMPOINT
TOPOINT >> 20DEC
CIRCLEPLUS >> 10INC DRAW
FROMPOINT >> 20INC

Clay> REMOTECONCENTRICCIRCLES >> MOVE CONCENTRICCIRCLES
Clay> CCTHING >> COLOR 5REMOTECONCENTRICCIRCLES
Meta> -CLEAN
Clay> CCTHING
Meta> -CLEAN
Clay> CCTHING
```

Figure #4: Concentric Circles



Exercise #3: Concentric Circle Thing

1. Enter the `RINGSANDTHINGS` region within your top level region.
2. Within your `RINGSANDTHINGS` region, define the `CCTHING`.
3. Do the afterglow thing, being sure to type the name `CCTHING` when prompted by the `-RECORD` meta command.
4. Move the file called `CCTHING.jpg` into the folder called `two` of your `website` folder.

3.4 The Dot, The Ring, and the Dot Ring

There is a command to increment the circle radius by 1 unit and a command to decrement the circle radius by 1 unit. These commands provide more flexability than the the previously introduced expand and shrink commands.

Text Segment #7: One pill makes you larger, and one pill makes you small

One way to paint a ring with a specific radius and a specific width is by repeatedly drawing rings of unit width, judiciously varying the radii. This idea is consistent with the theme in this text of playing with various sorts of building blocks. The first two lines of Clay Session #7 consitute the definitions of programs to simulate the pill that makes you larger and the pill that makes you smaller.

The next definition is for a command that paints a relatively large colored dot. The image show in Figure #5 (a) shows the image generated by running the program, assuming default condidtions.

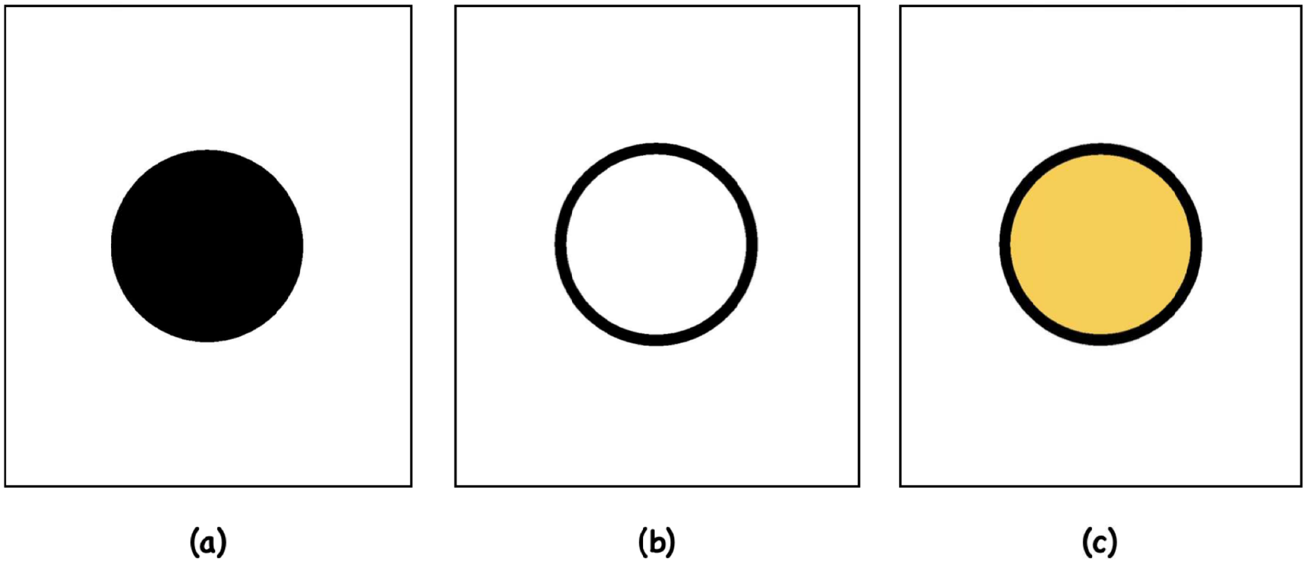
Then a program is written in the manner previously suggested to paint a relatively large ring of radius 20. Figure #5 (b) shows the result of running the program, assuming the default conditions.

The last bit of the session defines and runs a program to render a “ring dot” with a randomly colore interior and a black border. Figure #5 (c) shows the output of running this program.

Clay Session #7: The Dot and the Ring

```
Clay> BIGGERPILL >> X5 X2
Clay> SMALLERPILL >> S2 S5
Clay> THEDOT >> BIGGERPILL PAINT SMALLERPILL
Clay> THEDOT

Meta> -CLEAN
Clay> THERING >> BIGGERPILL OUTERRING INNERRING SMALLERPILL
Clay> OUTERRING >> 10INC+DRAW 10DEC
Clay> INNERRING >> 10DEC+DRAW 10INC
Clay> THERING

Meta> -CLEAN
Clay> THERINGDOT >> COLOR THEDOT BLACK THERING
Clay> THERINGDOT

Meta> -DISPLAY
THEDOT >> BIGGERPILL PAINT SMALLERPILL
BIGGERPILL >> X5 X2
SMALLERPILL >> S2 S5

Meta> -DISPLAY
THERING >> BIGGERPILL OUTERRING INNERRING SMALLERPILL
OUTERRING >> 10INC+DRAW 10DEC
INNERRING >> 10DEC+DRAW 10INC
BIGGERPILL >> X5 X2
SMALLERPILL >> S2 S5

Meta> -DISPLAY
THERINGDOT >> COLOR THEDOT BLACK THERING
THEDOT >> BIGGERPILL PAINT SMALLERPILL
BIGGERPILL >> X5 X2
SMALLERPILL >> S2 S5
THERING >> BIGGERPILL OUTERRING INNERRING SMALLERPILL
OUTERRING >> 10INC+DRAW 10DEC
INNERRING >> 10DEC+DRAW 10INC
```

Figure #5: The Dot and the Ring



Exercise #4: The Dot, The Ring, The Dot Ring

1. Enter the `RINGSANDTHINGS` region within your top level region.
2. Within your `RINGSANDTHINGS` region, define the `THEDOT`.
3. Within your `RINGSANDTHINGS` region, define the `THERING`.
4. Within your `RINGSANDTHINGS` region, define the `THERINGDOT`.
5. For the program called `THERINGDOT` program, please do the afterglow thing, being sure to type the name `THERINGDOT` when prompted by the `-RECORD` meta command.
6. Move the file called `THERINGDOT.jpg` into the folder called `two` of your `website` folder.

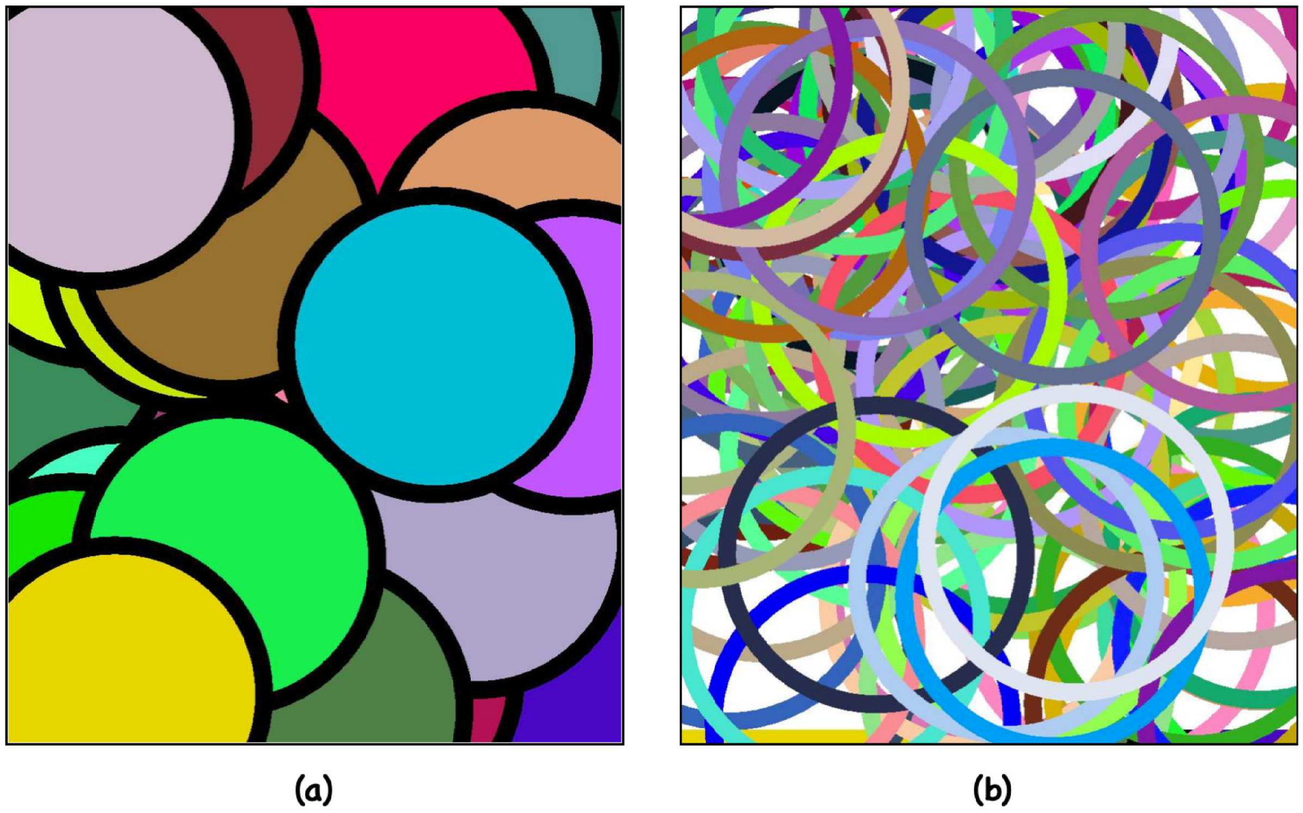
Text Segment #8: Two Images

The image in Figure #6 (a) was generated by running the program called `REMOTERINGDOTS` that is defined at the start of Clay Session #8. The image in Figure #6 (b) was generated by running the program called `REMOTECOLORRINGS` that is defined at about the half way mark of Clay Session #8.

Clay Session #8: The Dot and the Ring

```
Meta> -DISPLAY
Clay> REMOTERINGDOTS >> 100REMOTERINGDOT
Clay> REMOTERINGDOT >> MOVE THERINGDOT

Clay> REMOTERINGDOTS

Clay> REMOTECOLORRINGS >> 100REMOTECOLORRING
Clay> REMOTECOLORRING >> MOVE COLOR THERING

Clay> REMOTECOLORRINGS
```

Figure #6: Space Filling Programs



Exercise #5: Space Filling Programs

1. Enter the `RINGSANDTHINGS` region within your top level region.
2. Within your `RINGSANDTHINGS` region, define the `REMOTERINGDOTS` as presented in Clay Session #8.
3. Do the afterglow thing, being sure to type the name `REMOTERINGDOTS` when prompted by the `-RECORD` meta command.
4. Move the file called `REMOTERINGDOTS.jpg` into the folder called `two` of your `website` folder.

3.5 The Rest of the Exercises

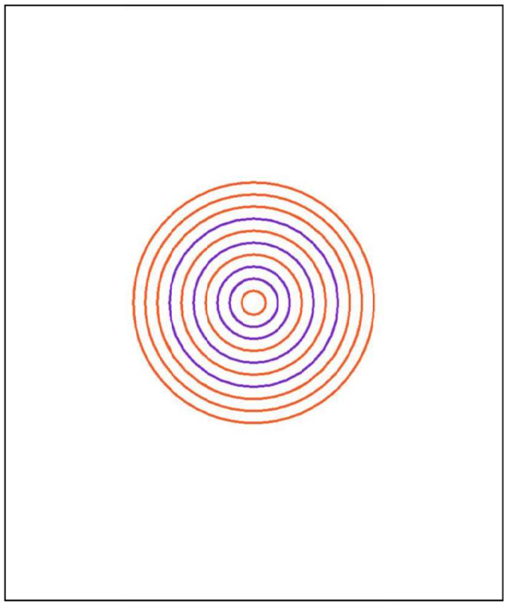
These exercises feature commands introduced in this chapter, the expand and shrink commands, the increment and decrement command, and the draw command. Naturally, this command set is supplimented by commands introduced in the previous chapter, the paint command, the move command, and the color changing commands. Furthermore, each of the exercises riffs on one or more of the programs presented in this chapter. And, of course, there will be the end of chapter web page building exercise!

Exercise #6: Prime Circles

Please do the following sequence of tasks, taking great care to attend to detail, expecially with respect to the naming and positioning of the file that you will be asked to save.

1. Run Gargole, click on the dot, and then enter the `RINGSANDTHINGS` region.
2. Write a program called `PRIMECIRCLES` which will generate the image shown in Figure #7. Specifically the program will draw orange circles of radii 1, 4, 6, 8, 9 and 10 times whatever the radius of the circle is when the program starts to run. It will also draw violet circles of radii 2, 3, 5 and 7 times whatever the radius of the circle is when the program starts to run. Please see the hints to the right of the image in Figure #7.
3. Do the 5 step afterglow thing, taking care to call your file `PRIMECIRCLES` when prompted.
4. Leave the region that you are working in by clicking on the `LEAVE` button. Then exit the system by clicking on the `EXIT` button.
5. Copy the file that you created by running the program, the file called `PRIMECIRCLES.jpg`, to the `two` folder within the `web` folder. You will be asked to do something with it before too long.

Figure #7: Prime Circles



This is a nice opportunity to make good use of the problem solving strategy of problem decomposition. The subproblems? (1) Draw the violet circles of raddi 2, 3, 5 and 7 times the radius of the circle at the time that the program starts to run. (2) Draw the orange circles of radii 1, 4, 6, 8, 9 and 10 times the radius of the dircle at the time the program starts to run. Your program should cleanly reflect this decomposition, and be written in terms of just two subprograms.

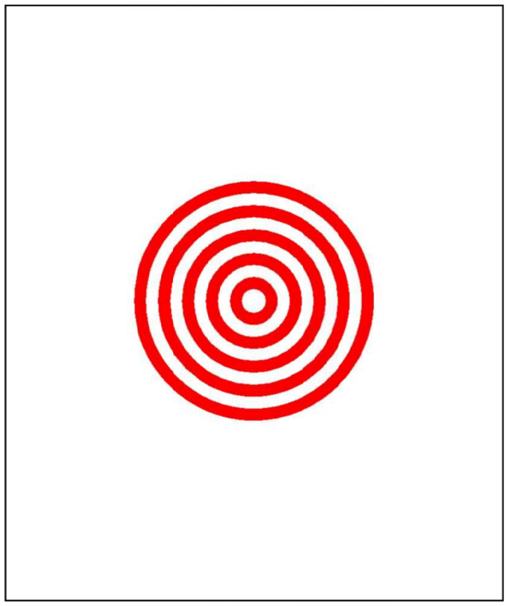
Exercise #7: Mega Target

Please do the following sequence of tasks, taking great care to attend to detail, expecially with respect to the naming and positioning of the file that you will be asked to save.

1. Run Gargole, click on the dot, and then enter the `RINGSANDTHINGS` region.
2. Write a program called `MEGATARGET` which will generate the image shown in Figure #8. Please see the hints to the right of the image in Figure #8.
3. Do the 5 step afterglow thing, taking care to call your file `MEGATARGET` when prompted.

4. Leave the region that you are working in by clicking on the LEAVE button. Then exit the system by clicking on the EXIT button.
5. Copy the file that you created by running the program, the file called MEGATARGET.jpg, to the two folder within the web folder. You will be asked to do something with it before too long.

Figure #8: Mega Target



You could generate this image by writing a rather long "one liner" — something like this:

```
MEGATARGET >> RED PAINT10 WHITE PAINT9 RED PAINT8 ...
```

But why don't you do something that is a bit more lengthy, something which makes explicit use of the problem decomposition strategy, again and again, and again? Why don't you complete this?

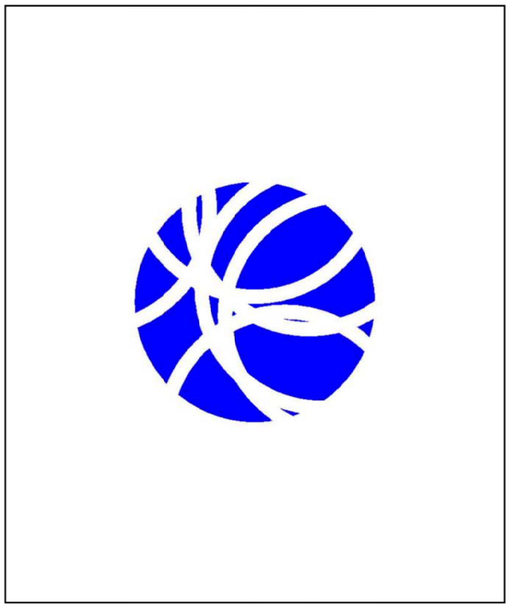
```
MEGATARGET >> RED PAINT10 MEGATARGET9
MEGATARGET9 >> WHITE PAINT9 MEGATARGET8
...
```

Exercise #8: Carving the Dot

Please do the following sequence of tasks, taking great care to attend to detail, expecially with respect to the naming and positioning of the file that you will be asked to save.

1. Run Gargole, click on the dot, and then enter the RINGSANDTHINGS region.
2. Write a program called BLUESPHERE which will generate the image shown in Figure #9. Please see the hints to the right of the image in Figure #9.
3. Do the 5 step afterglow thing, taking care to call your file BLUESPHERE when prompted.
4. Leave the region that you are working in by clicking on the LEAVE button. Then exit the system by clicking on the EXIT button.
5. Copy the file that you created by running the program, the file called BLUESPHERE.jpg, to the two folder within the web folder. You will be asked to do something with it before too long.

Figure #9: Blue Sphere



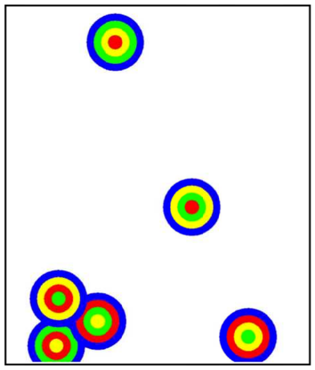
This is another nice opportunity to make good use of the problem solving strategy of problem decomposition. The subproblems? (1) Paint the blue dot. (2) Paint 10 randomly located white rings. The program should cleanly reflect this decomposition, and be written in terms of just two subprograms.

Exercise #9: Permutation Ring Things on Four Colors

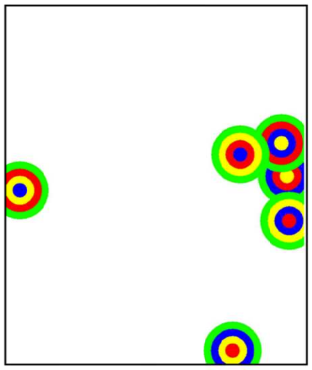
Please do the following sequence of tasks, taking great care to attend to detail, expecially with respect to the naming and positioning of the file that you will be asked to save.

1. Run Gargole, click on the dot, and then enter the `RINGSANDTHINGS` region.
2. Write a program called `BGRYPERMUTATIONS` which will generate the image shown in Figure #11. Please see the hints to the right of the image in Figure #11.
3. Do the 5 step afterglow thing, taking care to call your file `BGRYPERMUTATIONS` when prompted.
4. Leave the region that you are working in by clicking on the `LEAVE` button. Then exit the system by clicking on the `EXIT` button.
5. Copy the file that you created by running the program, the file called `BGRYPERMUTATIONS.jpg`, to the `two` folder within the `web` folder. You will be asked to do something with it before too long.

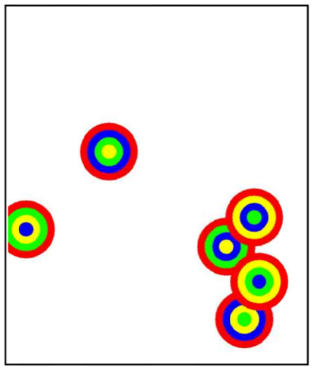
Figure #10: Four Sets of Six Permutations



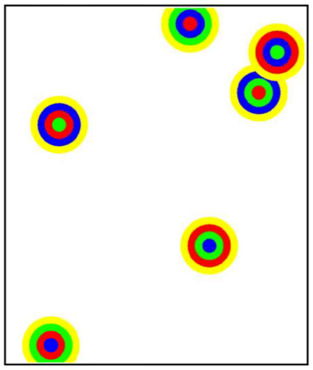
(a)



(b)

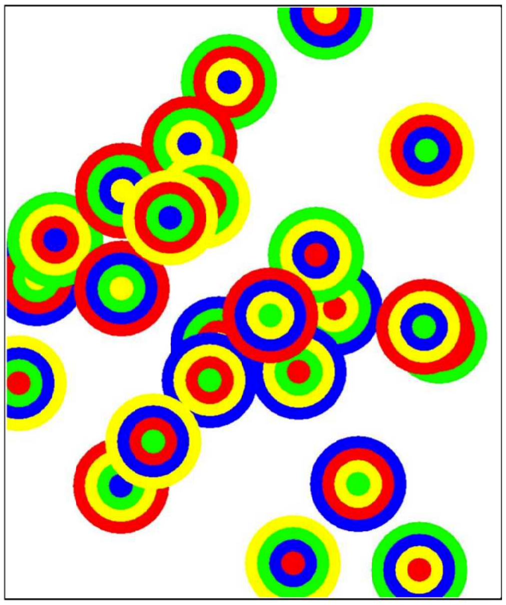


(c)



(d)

Figure #11: Twenty Four Permutations



If you would like to channel me, as I wrote this program, you would proceed in the following bottom up fassion:

(1) Write 6 programs to paint the permutation rings with blue on the outside. Perhaps you could call these BGRY, BGYR, BRGY, BRYG, BYGR, BYRG. Then write six programs to paint these rings in random locations. Perhaps you could call these PB1, PB2, PB3, PB4, PB5 and PB6. Write another 12 analogous programs for permutation rings with green on the outside. Another 12 for red on the outside. Another 12 with yellow on the outside.

(2) Write four programs to generate the images in the Figure #10, being sure to reference the low level programs appropriately. Perhaps you might call these four mid level programs BLUEPERMUTATIONS, GREENPERMUTATIONS, REDPERMUTATIONS, and YELLOWPERMUTATIONS.

(3) Write the top level program, BGRYPERMUTATIONS, which simply references the four midlevel programs.

Exercise #11: Web Page

Please do the following sequence of tasks, taking great care to attend to detail, expecially with respect to the naming and positioning of the file that you will be asked to save.

1. Run Gargole, click on the dot, and then enter the `LOTSOFDOTS` region.
2. Write a program called `NIGHTSKY` which will render 12 blue moons in a starry sky. An example run of the program that I wrote appears in Figure #8 (a).
3. Do the 5 step afterglow thing, taking care to call your file `NIGHTSKY` when prompted.
4. Leave the region that you are working in by clicking on the `LEAVE` button. Then exit the system by clicking

on the `EXIT` button.

5. Copy the file that you created containing the night sky image (emergent stars and explicit blue moons), the file called `NIGHTSKY.jpg`, to the `one` folder within the `web` folder. You will be asked to do something with it before too long.

Exercise #11: The Powerful Ideas

Do your best, based mainly on the text, but incorporating relevant information that you find via Google search, if you like, write a one paragraph description of each of the following powerful ideas. Do it by hand, in the space provided, if you are working from a hard copy of this text. Otherwise, do it in whatever reasonable place you like.

?

?

?

3.6 Clay Recap

This section reviews the Clay commands and the Meta commands that were used in this chapter, and it also summarizes the Clay constructs that were introduced.

Primitive Clay Commands and Constructs

The following list of commands is a complete list of the Clay primitive commands that were used in this chapter. It is ...

- ...

4 Deterministic Distributions and Particular Patterns

5 Hirst Dots and Variations

6 Things to do ...

1. Talk about miimalism in introduction.
2. Make a web page for this book.
3. Add image to web page one to Exercise 10 of chapter one.
4. Place the chapter 1 sample page out there for this book.
5. Select the quotes.
6. Proof read the text.
7. Spell check the text.
8. Write the Enter/Leave bit in chapter one.
9. Think about incorporating this stuff.
   (a) The Dot and the Ring
   (b) Lines of Dots
   (c) Movement Utilities
   (d) Lots of Dots
   (e) Hirst Dots
   (f) Erasing
   (g) BigLittleDots
   (h) StrangeRing
   (i) Dashes
10. Script for Video Introduction
    (a) Create `Paedia` folder.
    (b) Create `software` folder with `Paedia` folder.
    (c) Go to the `Paedia` page.
    (d) Download `Gargoyle` to the `Paedia/software` folder.
    (e) Run the Gargoyle program.
    (f) Enter the Dotsworld.
    (g) PAINT -CLEAN
    (h) Define: BIGGERDOT
    (i) Define: SMALLERDOT

(j) Define: RINGTHING

(k) Run: RINGTHING

(l) Record: RINGTHING

(m) Display: RINGTHING

(n) Exit

(o) Open RINGTHING.jpg by double clicking

(p) Return to the `Paedia` page

(q) Visit the model Lots of Dots page.

(r) Visit the text.

(s) Visit this invitation.

(t) Issue the invitation: Read through the text, slowly, doing the exercises as you go. Maybe you will enjoy the experience. Maybe it will change your relationship to computation in a positive way!