Fractals and Lite–Brite: Coloring a Grid

Peter Andrews
Andrew Mertz
Nancy Van Cleave

Mathematics and Computer Science Department
Eastern Illinois University

September 24–25, 2010

Overview

- Introduction
- Why is this nifty?
- Tiling the Graphics Window
- The Julia Set for CS I
- The Lite–Brite Assignment for CS II
- Results & Contact Information

Introduction

- Concept: Using canvas as grid of blocks to be colored according to some rule(s)
  - Checkerboard
  - Julia Set – fractal-based coloring
  - Lite–Brite – event-driven programming
- Offers Variety:
  - Colorful (interactive) graphics
  - Levels of difficulty
  - Language features emphasized
- Incremental, Extensible

Why Is This Nifty?

- Fits the course framework nicely :
  - Nested loops, checkerboard, etc.
  - A natural extension of work already completed
- General interest and appeal :
  - Julia Sets can be spectacular - students enjoy experimenting
  - Their own Lite–Brite program provides interactive fun and makes them proud owners

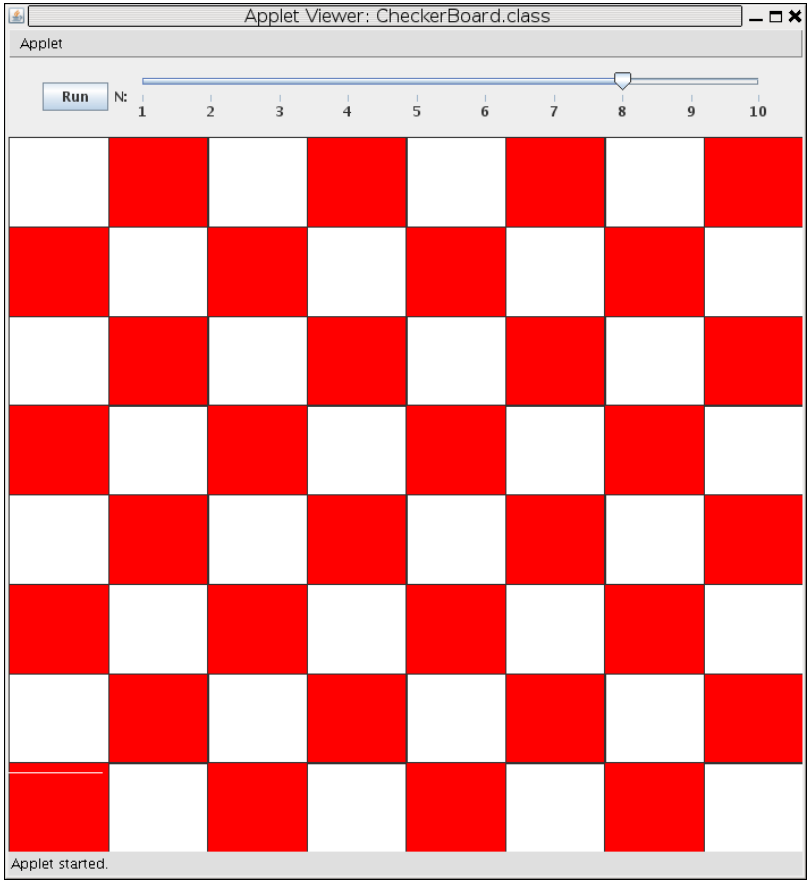
Why It's Nifty – Continued...

- No special background needed :
  - Underlying theme: tiling the window with blocks
  - Julia Sets: students are given skeleton program (includes coloring method)
  - Lite–Brite: a straight–forward assignment, simple to understand, students know when they are successful
- Surprises for students :
  - **Wow**! I can actually make a computer do this?
  - I wrote a program that I enjoy running!
- Provides a springboard to topics often not seen until later

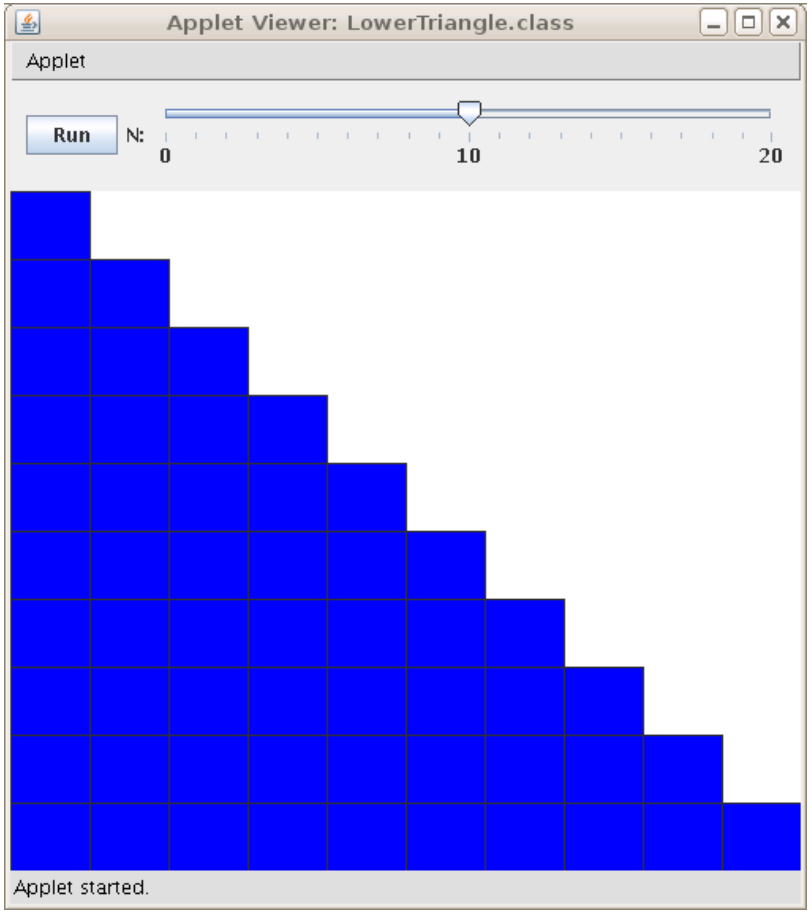
EIU: First Programming Course

- ACM Java Library
- Graphics from Week 1
- Nested loops in Week 4
- Slider and DualSlider programs are provided
- Incremental difficulty

Tiling Images I

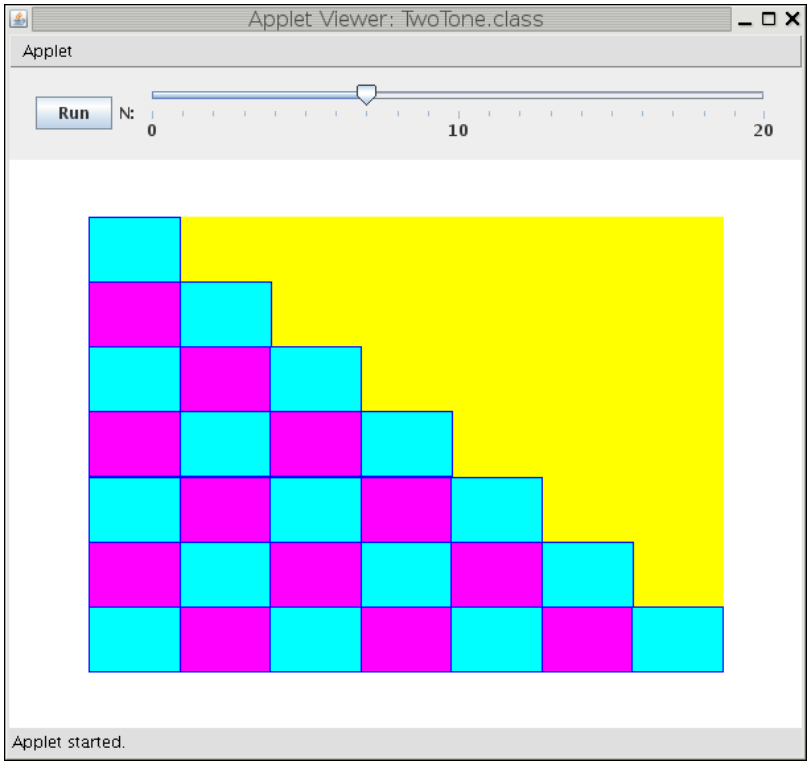


Checker Board

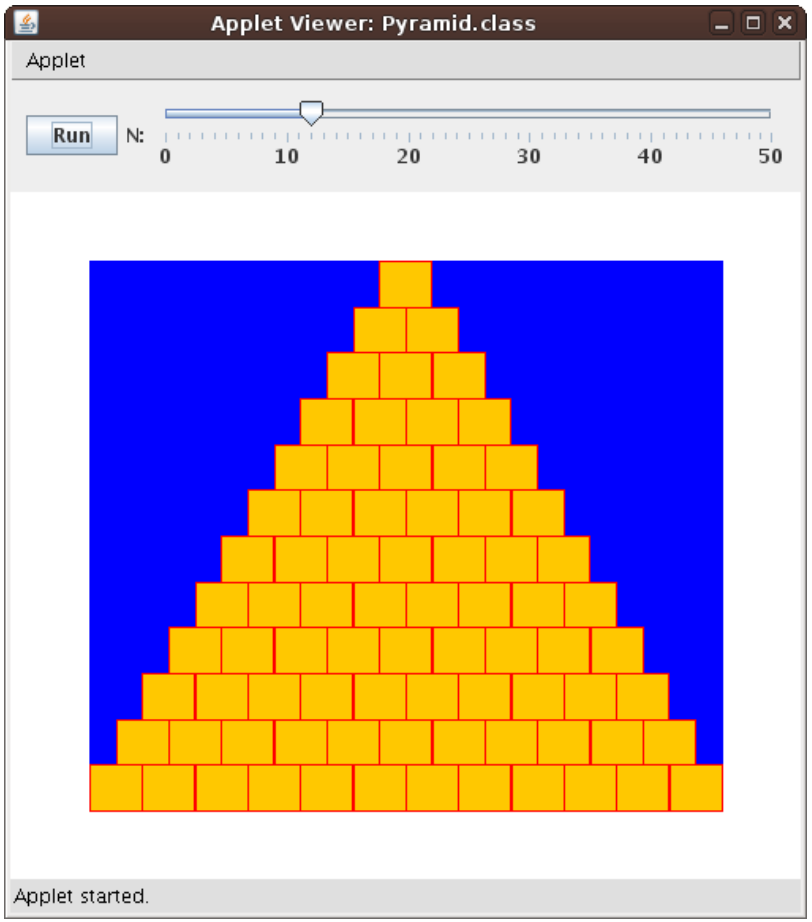


Lower-Left Triangle

Tiling Images II

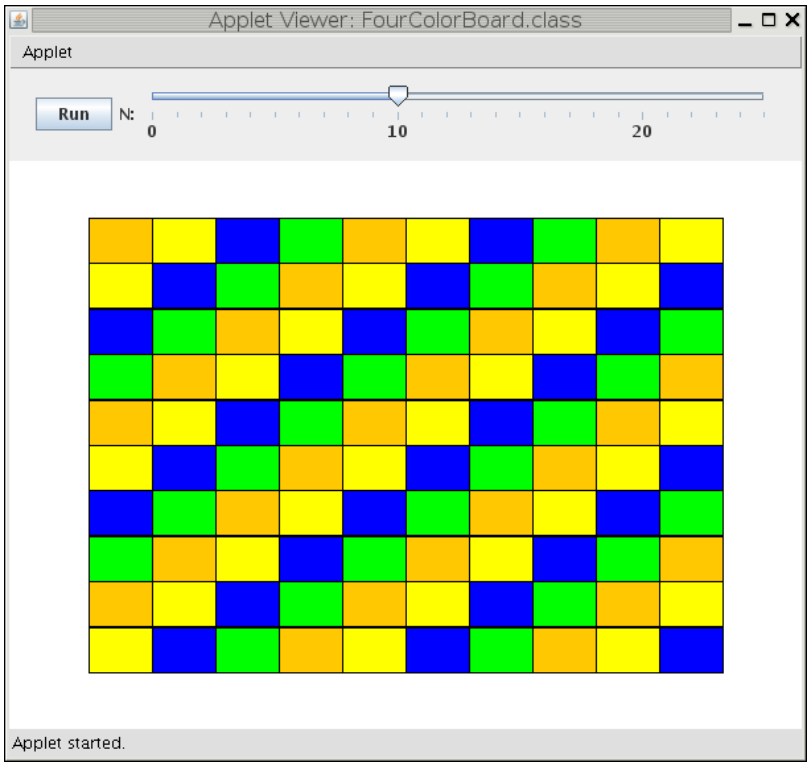


Two-Tone Triangle

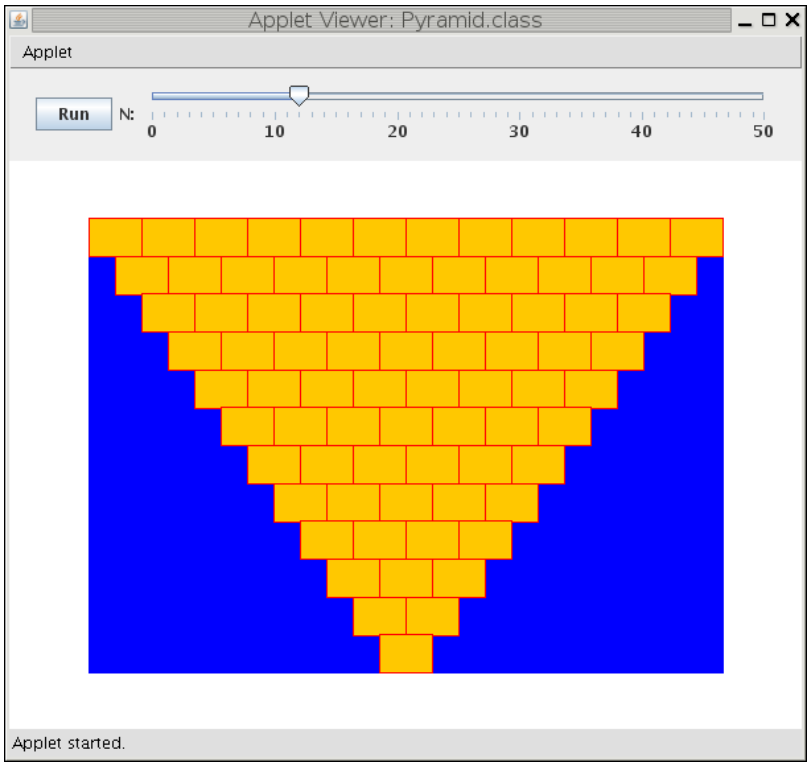


Pyramid

Tiling Images III



Four-color Tiling



Reverse Pyramid

The Julia Set

To every real–valued pair, (a, b), we can associate a function of two variables — referred to as the Julia map for (a, b) — denoted by $F_{a,b}$, and described by the formula:

$$F_{a,b}(x, y) = (x^2 - y^2 + a, 2xy + b)$$

Repeatedly applying the function $F_{a,b}$ to successive results:

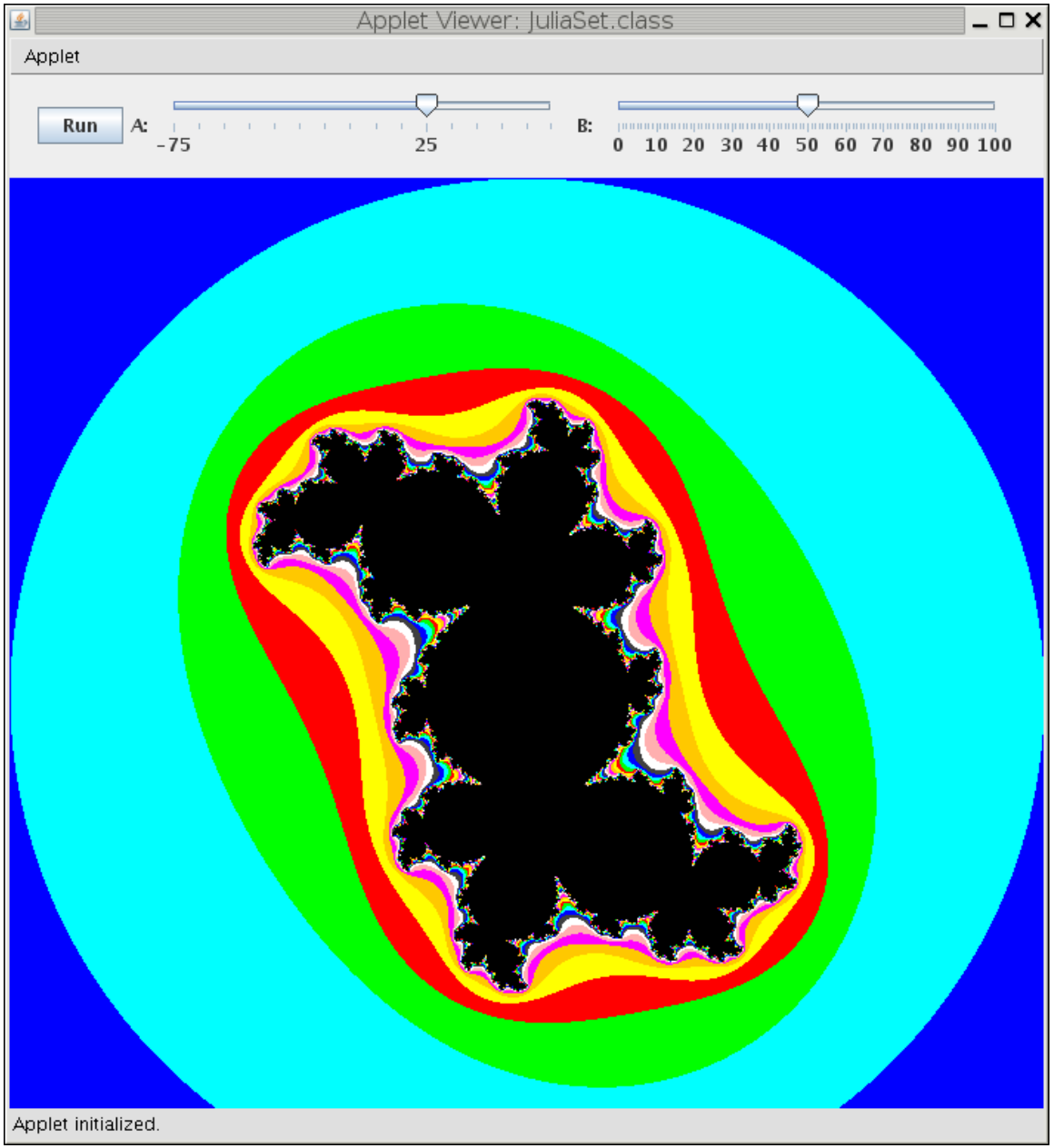
$$\begin{aligned} F_{a,b}(x, y) &= (x_1, y_1) \\ F_{a,b}(x_1, y_1) &= (x_2, y_2) \\ &\vdots \end{aligned}$$

will either produce points near the original (x, y) or after some number of iterations will cross a threshold and move away.

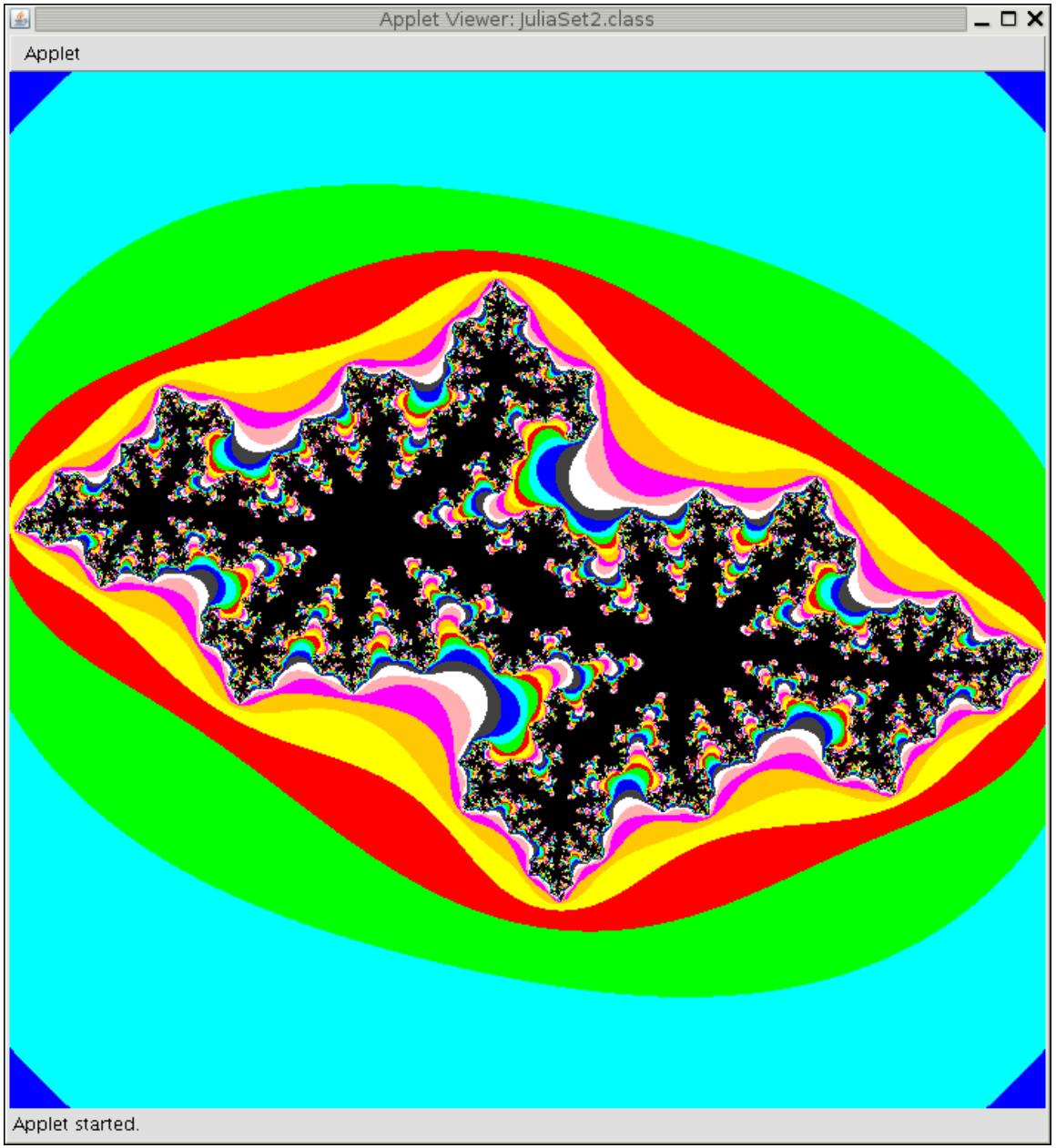
The Julia Set

- The Julia set for (a, b), $J_{a,b}$, is the collection of all points in the plane from which you can start and never get too far away from the original by repeated iterations of $F_{a,b}$.
- The points that don't get out within a certain, preset number of iterations (those in the Julia set) are colored **black**.
- One way to display starting values that *do* escape is to **color** them according to **how many iterations** it takes to get outside a threshold circle.
- Different choices of (a, b) often give rise to quite exotic pictures.

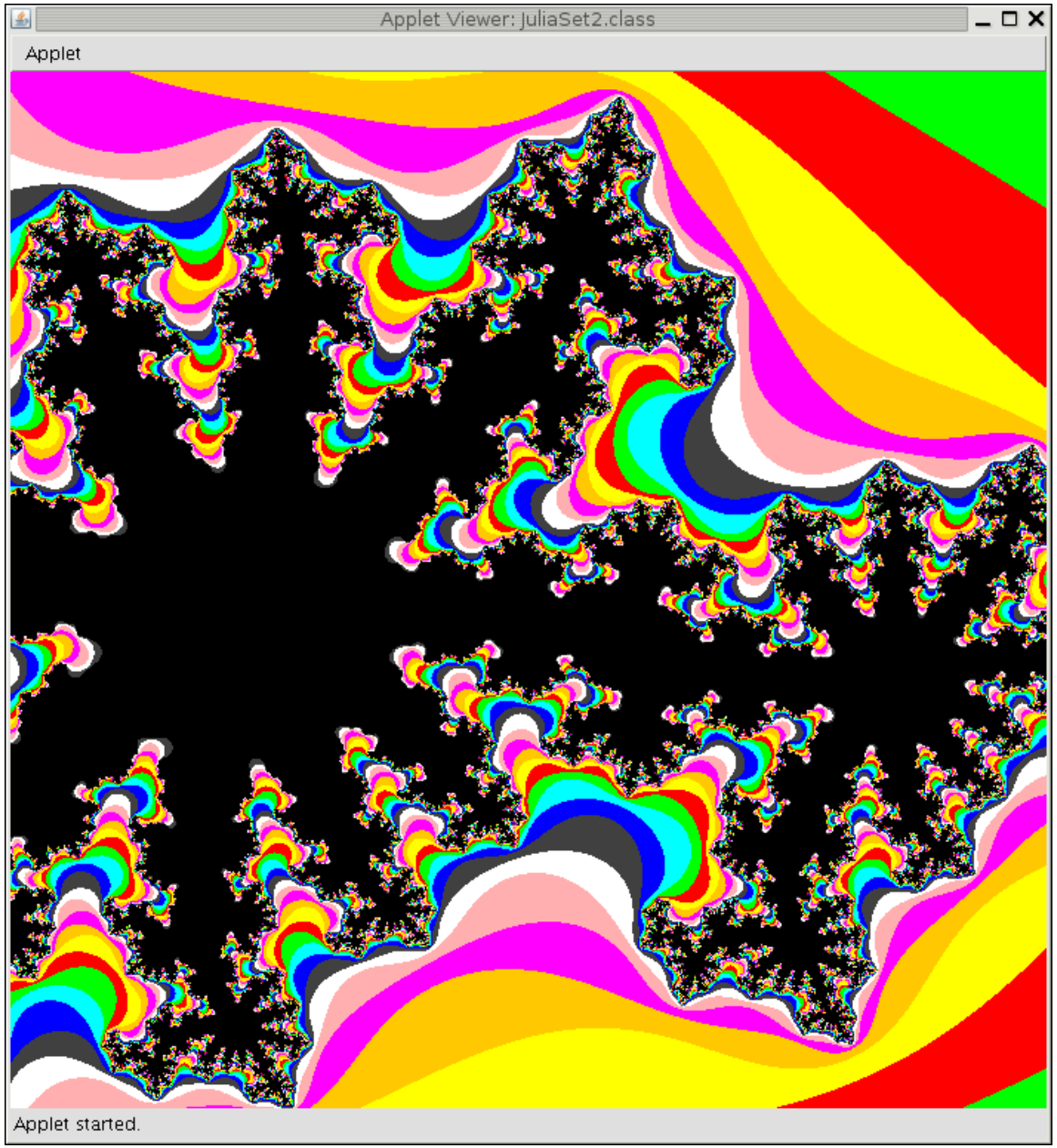
Julia Set Images — 1



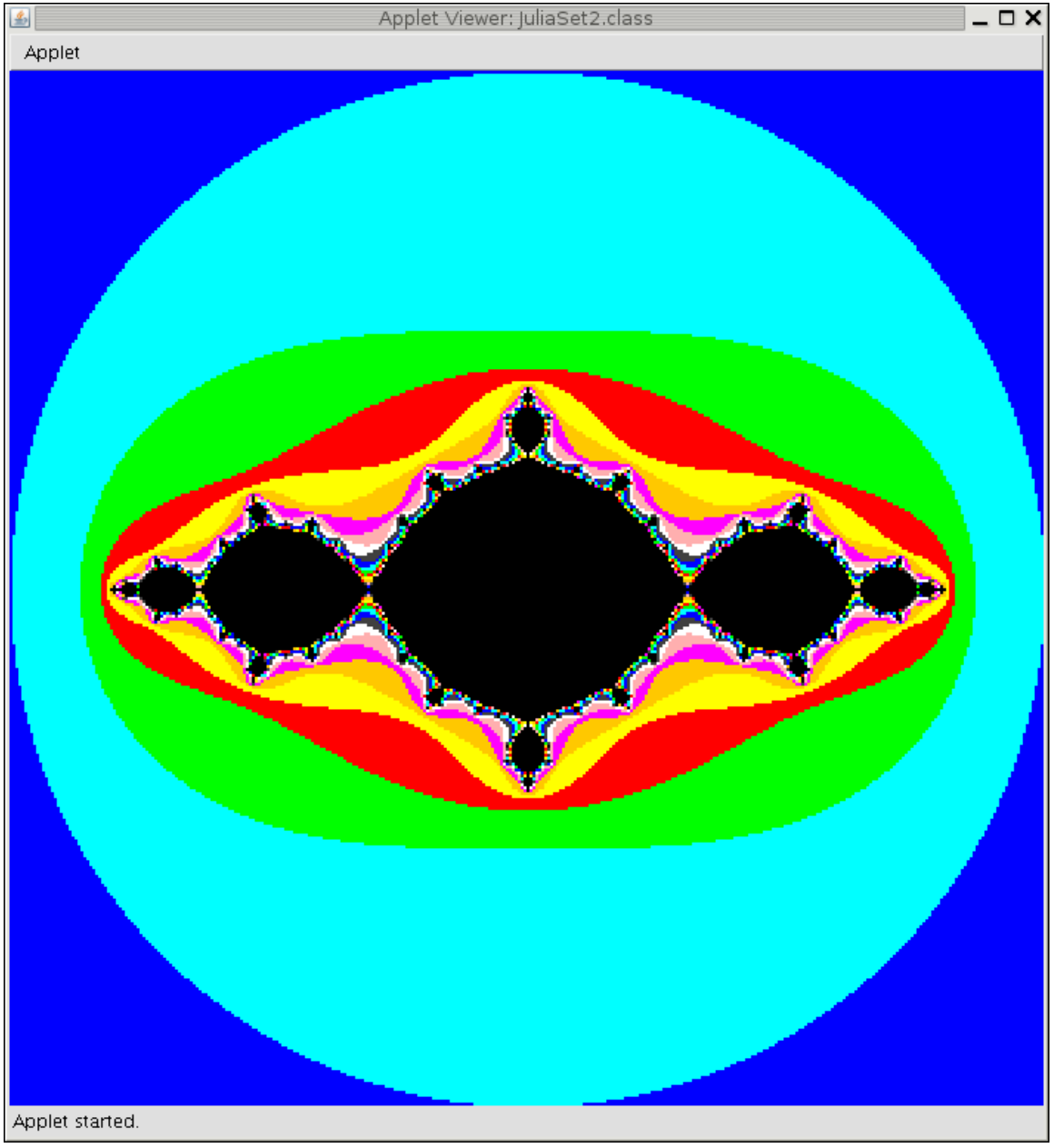
Julia Set Images — 2



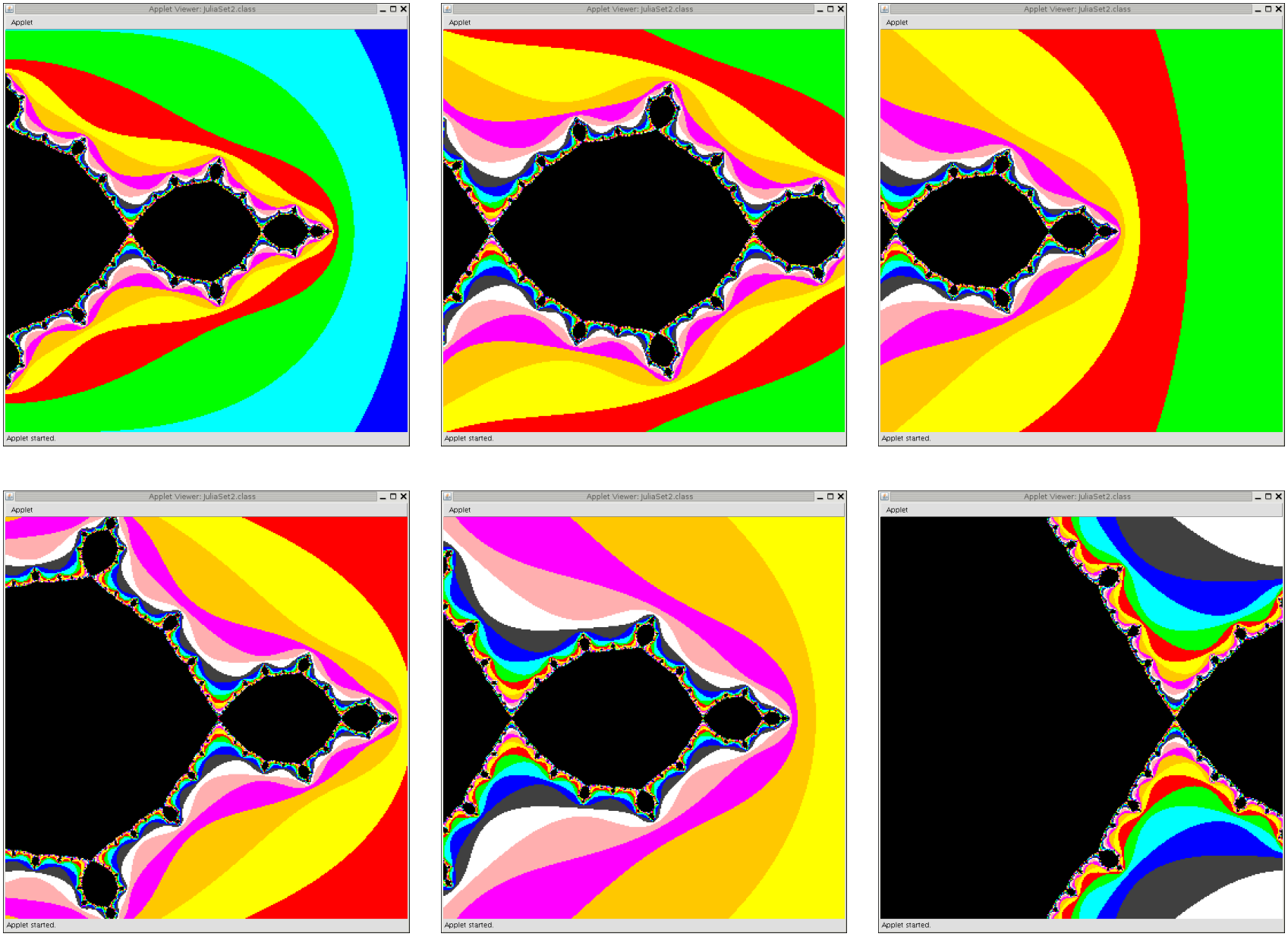
Julia Set Images — 2 Zoom



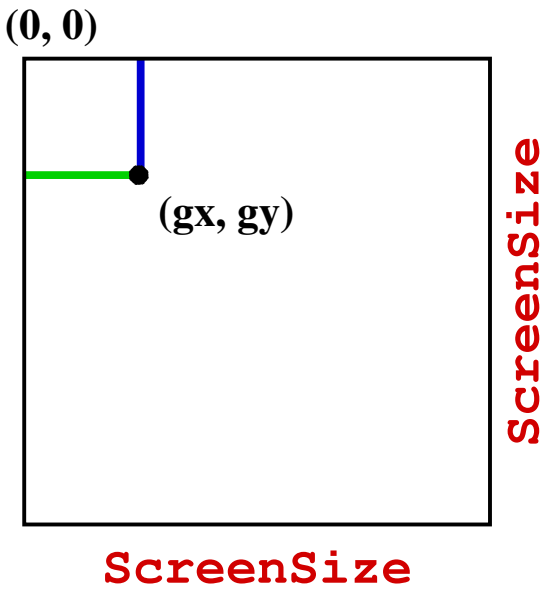
Julia Set Images — 3



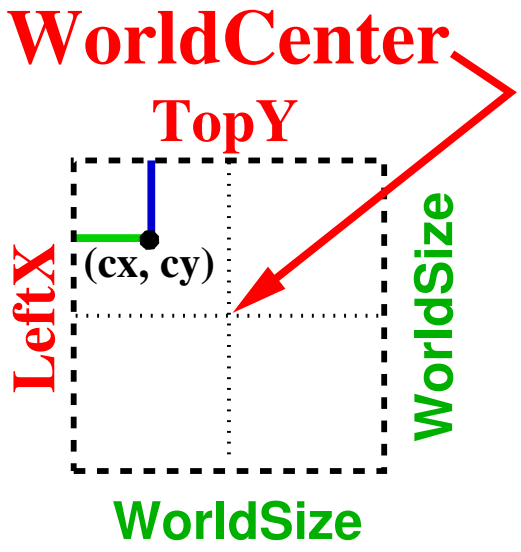
Julia Set Images — 3 Shift & Zoom



The Julia Set – Scaling from Graphics to "Real World"



graphics
window



"Real World"
Cartesian
plane

The Julia Set Assignment
For each grid block in the graphics window:

1. Find its upper left corner position (complete **BlockCorner()**)
2. Find the corresponding point in the world region (complete **ScreenToWorld()**)
3. Find the color this point (block) should get (**JuliaColor()**, provided)
4. Draw a block of the correct size, location, and color to represent the Julia map

The finer the mesh of squares covering the graphics window, the higher the resolution and smoother the curves.

The Julia Set: constants

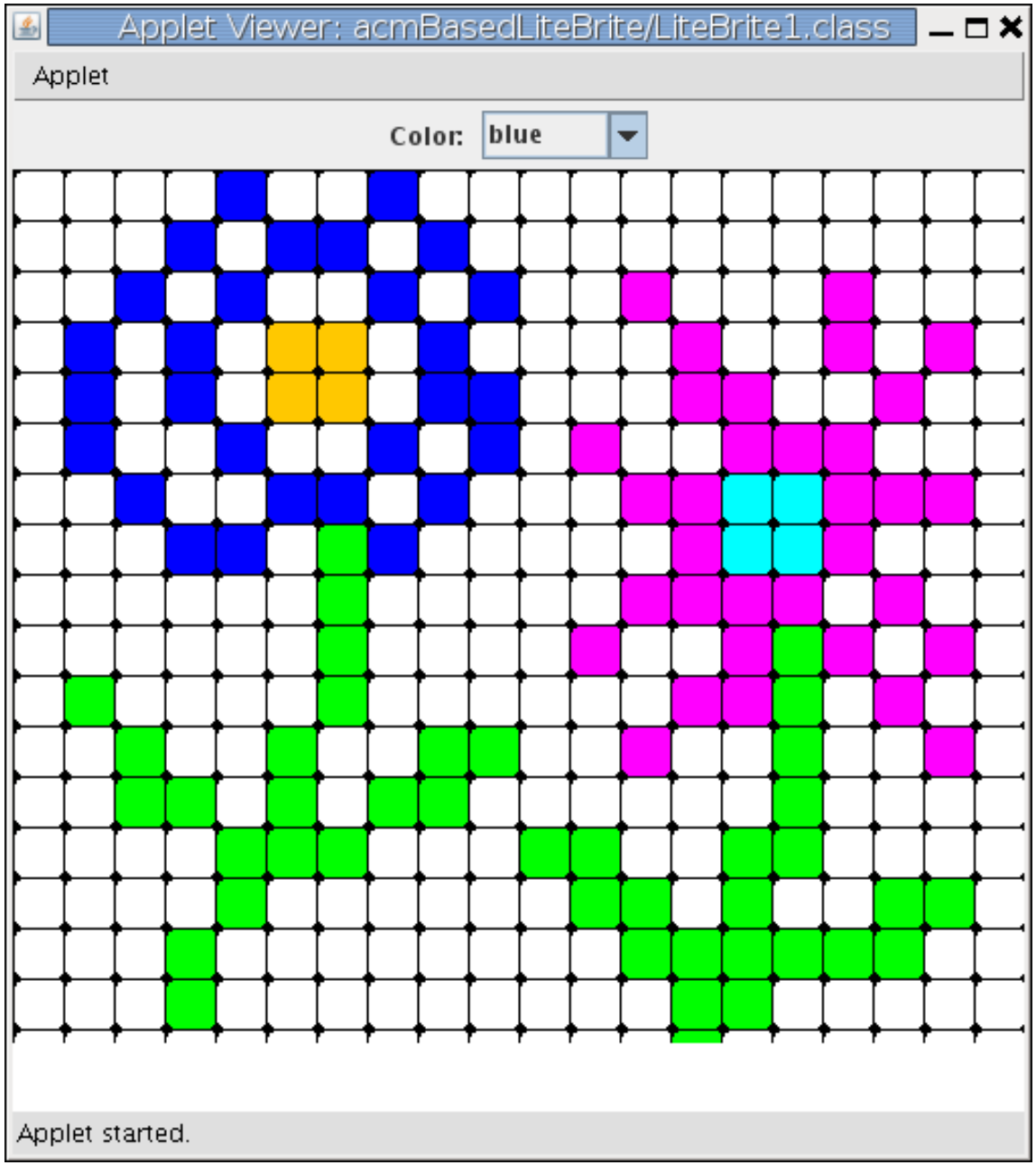
ScreenSize	700	Size of graphics window (pixels)
WorldSize	4.0	Size of "window" in real world
WorldCenter	(0.0, 0.0)	Center of real world
GridSize	700	Number of blocks per row/col *can use 350 for 2x2 blocks
MaxColors	11	Number of colors used
MaxIterations	40	Time given to attempt escape
Threshold	2.0	Distance from beyond which a point will not return

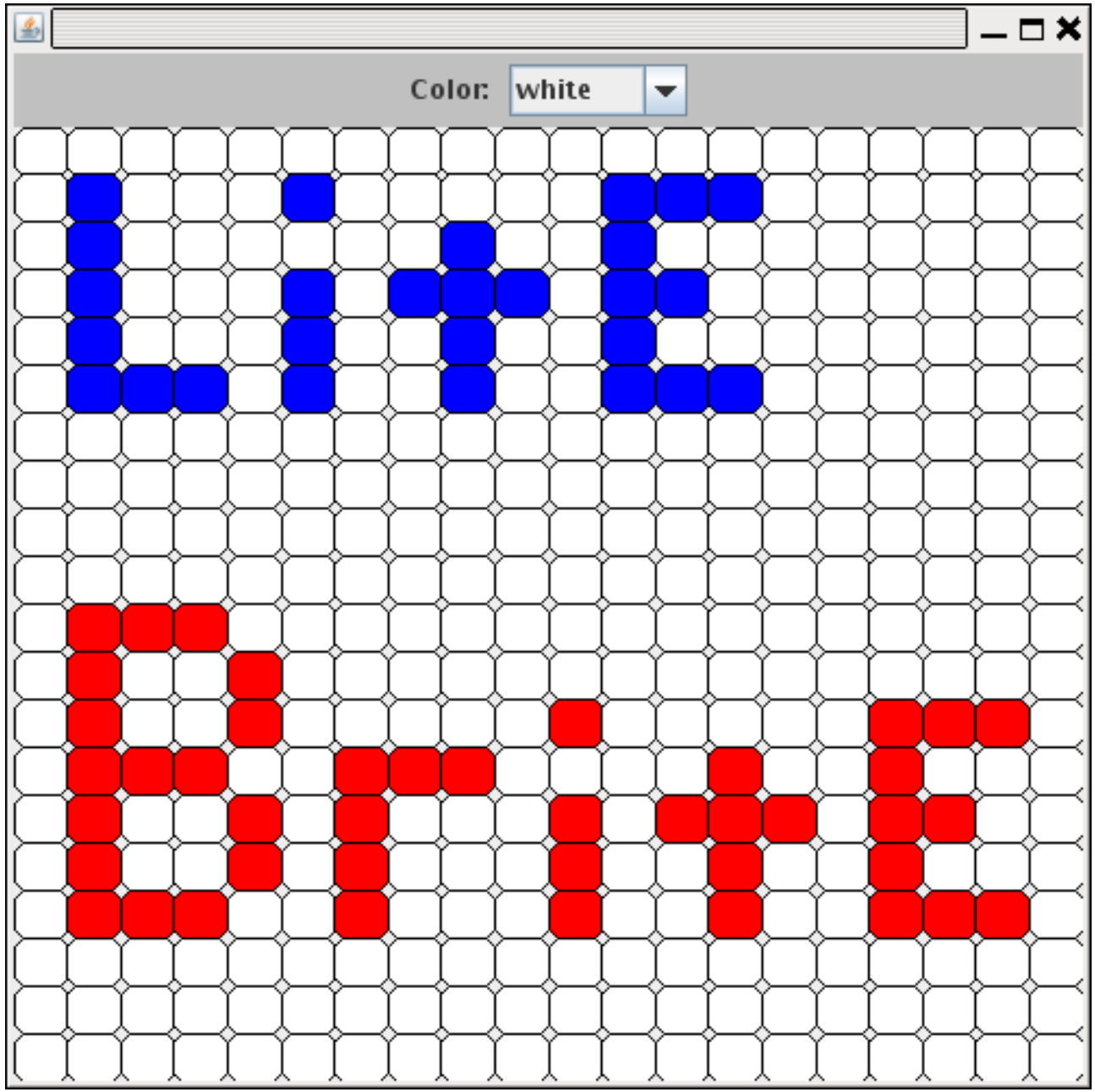
The Lite–Brite Assignment for CS II

Remember the Lite–Brite toy from the 70's?
You placed black paper over a lighted grid and pushed colored pegs though it to create pictures.

Common Threads

- User can choose a color via a combo box or other GUI elements
- The majority of the window consists of a grid of colorable "blocks"
- Whenever the mouse button is pressed on a block or the mouse is dragged over a block its color changes to match the currently selected color


Applet Viewer: acmBasedLiteBrite/LiteBrite1.class
Applet
Color:
blue
Applet started.


Color:
white

Many Solutions

There are many ways to solve this problem which gives instructors lots of flexibility in how and when it is used.

We will present three different solutions:

acm based Uses the acm library for graphics and to slightly simplify the event handling

swing painting Uses a list to hold the colors for each block and overrides `paintComponent` to draw them

swing layout Uses the grid layout manager to create a grid of colorable panels

Live Demo

Source code for lite brite can be found at
`www.eiu.edu/~mathcs/`

Extendable

The basic lite brite assessment is simple (can be solved in approximately 100 lines of code).

At the same time it can easily be extended to become more complex and to exercise other skills.

- Loading and saving images
- Printing
- Saving bitmap images
- Zooming, rotating, creating negative images, or other image manipulations.

Results

- Julia Set:
  - Advanced students noticed CS I displays and wanted to write program too
  - (Some) Students enjoyed modifying Julia Set program: new colors, different world size or center
- Lite Brite:
  - (Some) Students would become engrossed in playing with their own Lite Brite programs
  - Lite Brite was considered more "fun" than Julia Set
- In both assignments, students learned a great deal

Contact Information

Peter Andrews, pgandrews@eiu.edu

Andrew Mertz, aemertz@eiu.edu

Nancy Van Cleave, nkvancleave@eiu.edu

URL: www.eiu.edu/~mathcs

Thank You