FANG Project

# Purpose

These notes capture my effort to recreate the FANG Java Game Engine as described in the book ***Introductory Programming with Simple Games***. I recently picked up the book to read as part of the CRVG project. The book was published in 2011, which at the time of this writing is over 14 years old. The book and software were part of an effort during that time to teach programming by combining it with learning to make computer games. Many colleges merged the two efforts in the course offerings to attract more young people to the computer science field. In fact, many colleges offered specialized majors in game development that spanned programming, game design and creating art assets.

The use of Java as the first programming language to teach computer science has seen a large decline since this book was written and the approach used in the book to teach programming via video game creation did not become as popular as the proponents hoped. The college programs evolved to utilize C++, a more natural language for creating games or dropped the notion of teaching game creation and programming at the same time. In addition, in many computer science curriculums Java has been replaced as the first programming language with other languages such as, Python, JavaScript. The trend away from Java seems natural since everyone can see the trends in the last decade has been towards data science, AI, front-end web development, using nodejs and other tools.

The decline in the idea of teaching Java using the creation of games as a gimmick is probably the reason the FANG Java Game Engine used in the book is almost impossible to find on the Internet today. I decided to take it on as a challenge to try to recreate FANG Java Game Engine so that if anyone gets a hold of the book and wants to be able to follow the examples, they will be able to use my version. I plan on following the same package names and methods that I find in reading the book.

The original project was an open source project, but it appears no one saved the source code on GitHub so it can be readily obtained. Maybe the source code is hiding somewhere in GitHub, but I could not find it. I managed to find fang2 but not all the other packages (e.g. spacesprites). We will mainly support running the programs in the textbook as java applications. Java applets no longer run on browsers without some effort on the part of the user to configure the browser to allow them to run. What is even tougher is that the last Java version to support applets was Java version 8, it was deprecated starting with Java 9 and completely removed in Java 17!

I had to install Java 8 on my machine and by 2030 there will be 0 support. I plan on building a version of fang for Java 17 and Java 21. Since I plan on making all the code available anyone should be able to create a jar for any target version of Java they desire.

Note: Game names, game company names, and video game program names are the property of their respective trademark or copyright holders.

# My Approach

I eventually plan on learning how to use JavaFX and maybe FXGL (<https://github.com/AlmasB/FXGL/wiki/Introduction>) using the same package name and methods supplied by the original FANG Java Engine implementation. I may originally utilize a combination of AWT and Swing libraries, since it appears that was the approach used in the original code before I try to create a version with the latest Java UI library.

I will primarily only work with the Javadoc material so that my implementation remains my own.

# How Things Work

## Case: Empty Code

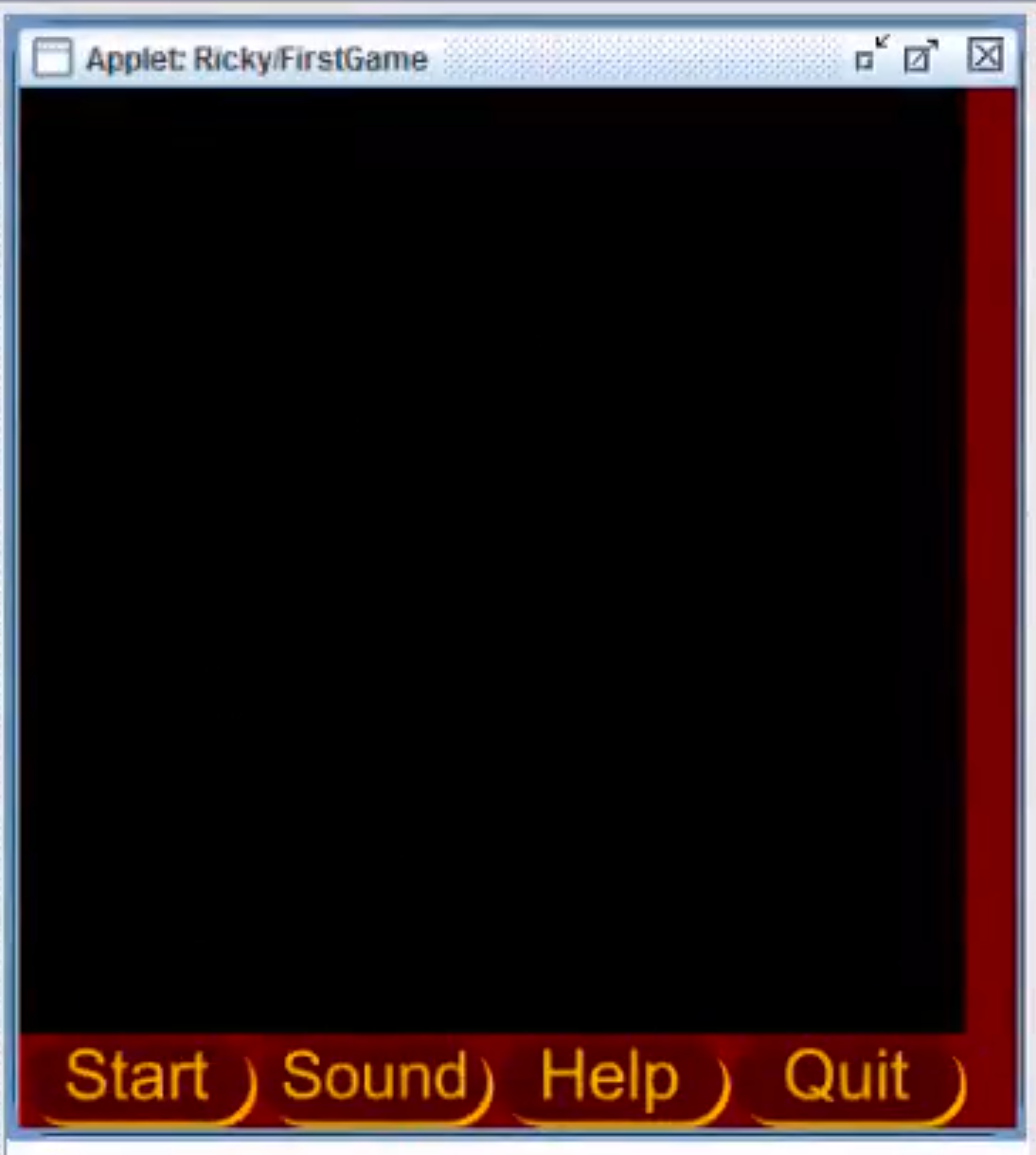
Suppose I created the project: Ricky/FirstGame

The code to display the default/empty application is:

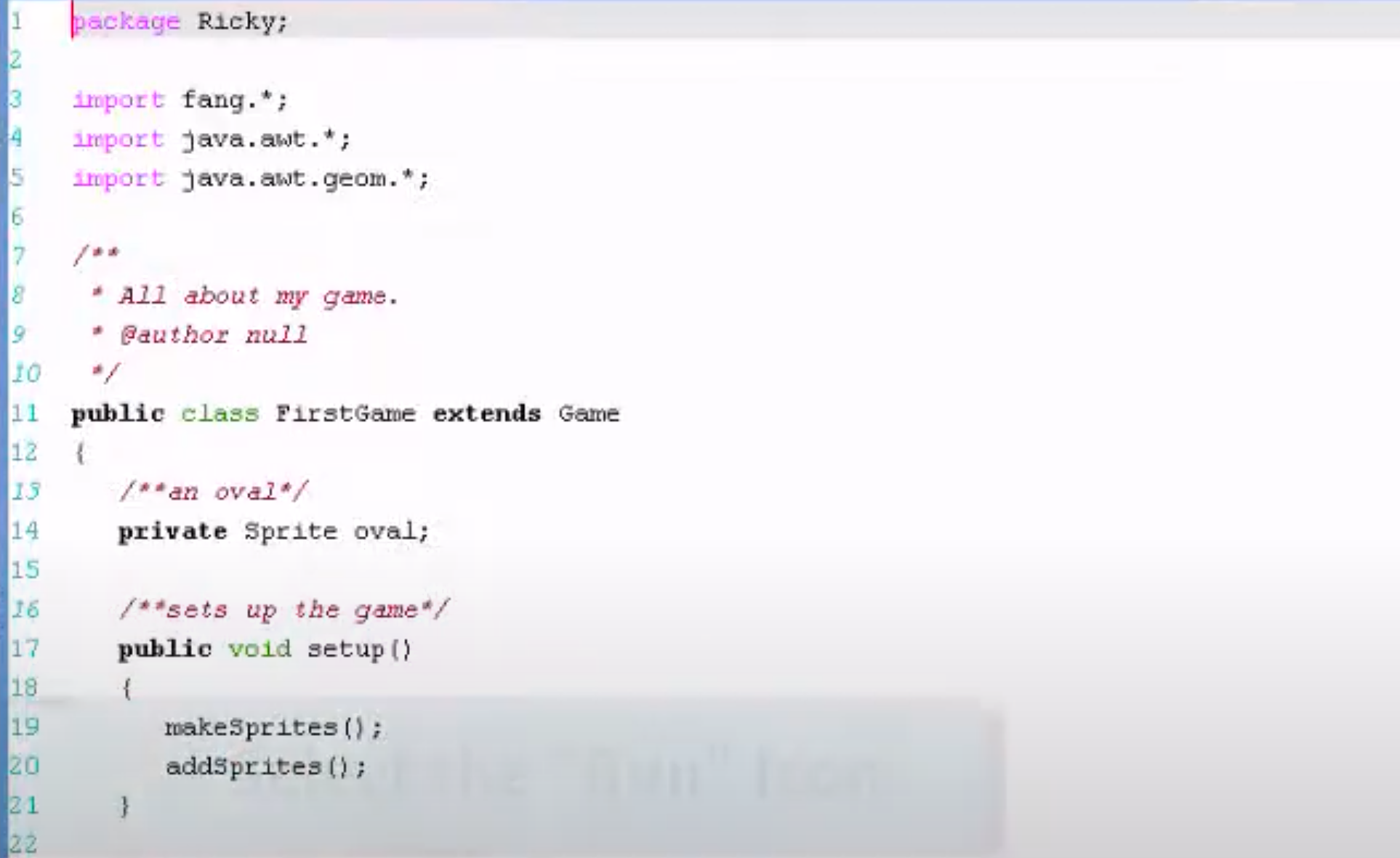
A screen shot of a computer

AI-generated content may be incorrect.

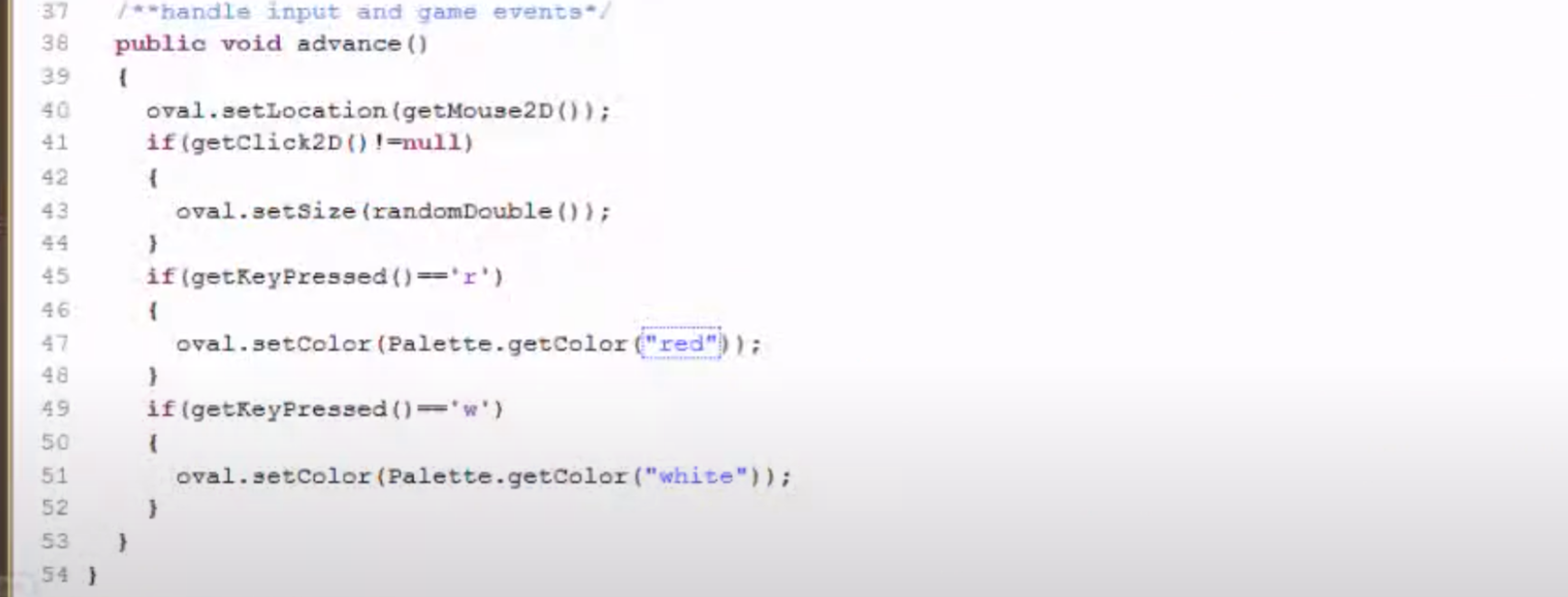
Note, the above was created as an Applet.



## Case: Simple Game #1









# Javadoc

fang

fang.attributes

fang.code

fang.experimental

fang.media

fang.network

fang.novice.transformers

fang.sprites

fang.transformers

fang.ui

fang.util

info.gridworld

info.gridworld.actor

xx

# Wackadot

1. package wackadot;

2.

3. import fang2.core.\*;

4. import fang2.sprites.\*;

5. import java.awt.\*;

6. import java.awt.geom.\*;

7. /\*\*

8. \* Use mouse to wack dots and move them.

9. \* Score points for wacking dots

10. \*

11. \* @author kris pepper

12. \* @version 1

13. \*/

14. public class Wackadot extends GameLoop

15. {

16. // note that GameLoop contains the variable canvas

17. // it also has a variable called random

18. private Sprite dot;

19. private Sprite redDot;

20. private Sprite blueDot;

21. private StringSprite scoreSprite;

22. private int redScore, blueScore;

23.

24.

25. public static void main(String[] args)

26. {

27. Wackadot mygame = new Wackadot();

28. mygame.runAsApplication();

29. }

30. @Override

31. public void startGame()

32. {

33. makeSprites();

34. addSprites();

35. redScore = 0;

36. blueScore = 0;

37. }

38. @Override

39. public void advanceFrame(double timePassed)

40. {

41. Point2D.Double mouse = getPlayer().getMouse().getLocation();

42. dot.setLocation(mouse);

43. handleCollisions();

44. }

45. public void makeSprites()

46. {

47. dot = new OvalSprite(1,1);

48. dot.setScale(.1);

49. dot.setLocation(.5,.5);

50. dot.setColor(Color.RED);

51.

52. redDot = new OvalSprite(1,1);

53. redDot.setScale(.1);

54. redDot.setLocation(

55. random.nextDouble(),

56. random.nextDouble());

57. redDot.setColor(Color.RED);

58.

59. blueDot = new OvalSprite(1,1);

60. blueDot.setScale(.1);

61. blueDot.setLocation(

62. random.nextDouble(),

63. random.nextDouble());

64. blueDot.setColor(Color.BLUE);

65.

66. scoreSprite=new StringSprite("Red Score: " + redScore + " Blue Score: "

67. + blueScore);

68. scoreSprite.setHeight(0.08);

69. scoreSprite.rightJustify();

70. scoreSprite.topJustify();

71. scoreSprite.setLocation(1, 0);

72.

73. }

74. public void addSprites()

75. {

76. canvas.addSprite(dot);

77. canvas.addSprite(redDot);

78. canvas.addSprite(blueDot);

79. canvas.addSprite(scoreSprite);

80. }

81.

82. private void handleCollisions()

83. {

84. if (dot.intersects(blueDot))

85. {

86. blueDot.setLocation(

87. random.nextDouble(),

88. random.nextDouble());

89. blueScore++;

90. scoreSprite.setText("Red Score: " + redScore + " Blue Score: "

91. + blueScore);

92. }

93. if (dot.intersects(redDot))

94. {

95. redDot.setLocation(

96. random.nextDouble(),

97. random.nextDouble());

98. redScore++;

99. scoreSprite.setText("Red Score: " + redScore + " Blue Score: "

100. + blueScore);

101. }

102.

103. }

104. }

105.