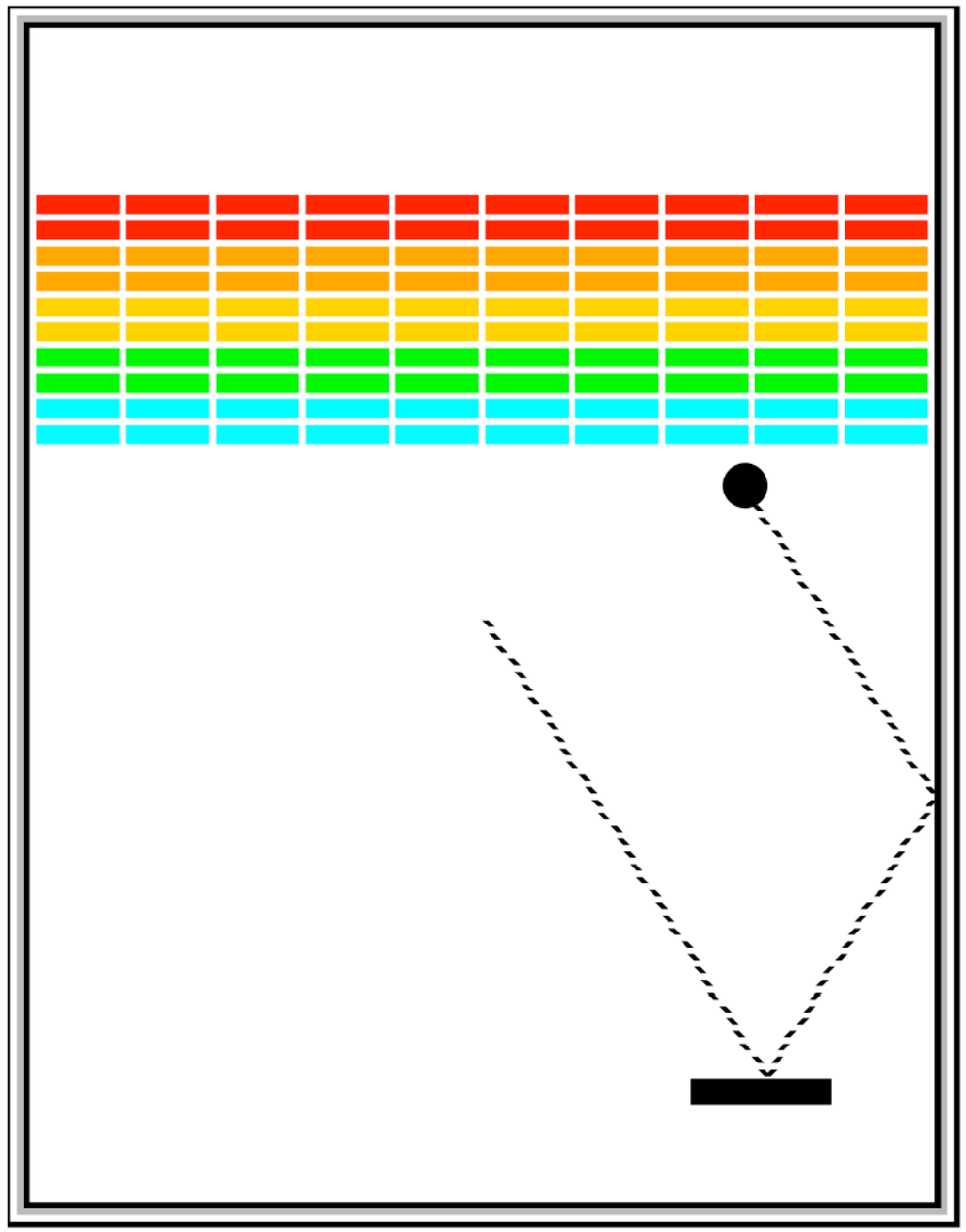
YEAH session #3



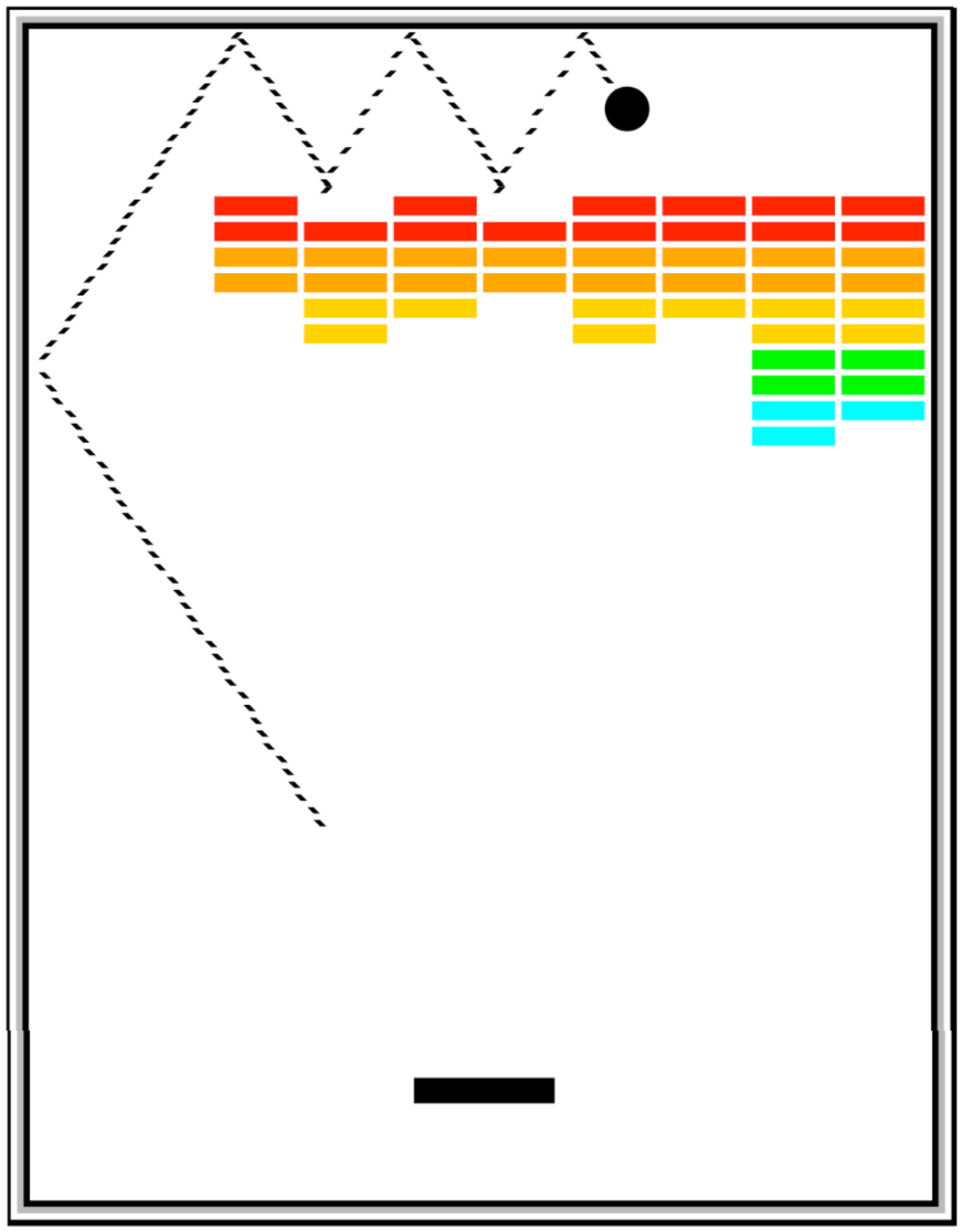
4 February 2013, 7:30p-8:30p

Miles Seiver

Review session schedule

Topic	Date	Time	Location
assignment 3	today!	now!	here!
midterm 1	Sun 9 Feb	1p - 4p	Hewlett 200
assignment 4	Thu 13 Feb	5:30p - 6:30p	Hewlett 200
assignment 5	Sun 23 Feb	7p - 8p	Hewlett 200
midterm 2	Sun 2 Mar	1p - 3p	Hewlett 200
assignment 6	Thu 6 Mar	5:30p - 6:30p	Hewlett 200
assignment 7	Sun 16 Mar	7p - 8p	Hewlett 200

Assignment 3



- Breakout
- Due Monday, February 10 at 3:15pm

Variable scoping

```
for (int i = 0; i < 5; i++) {
    int y = i * 4;
}
i = 3; // Error!
y = 2; // Error!
```

Variable scoping cont.

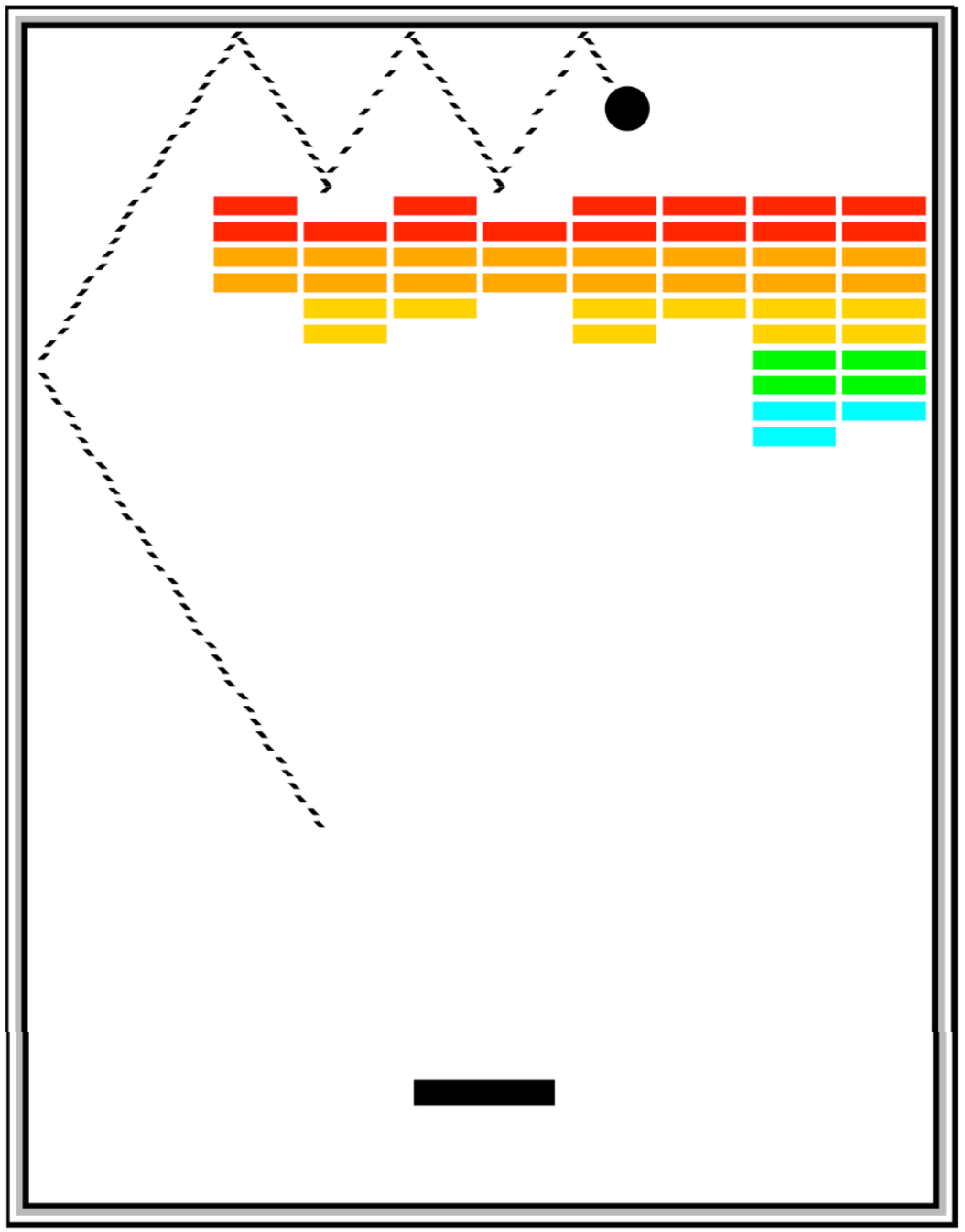
```
public void run() {
    int x = 5;
    someOtherMethod();
}
private void someOtherMethod() {
    x = 4; // Error!
}
```

Many returns

```
private int thisIsLegal(int x)  {
        if (x == 5)  {
                return 0;
        }
        return 1;
}
```

The only way we can get here is if x is not equal to 5.

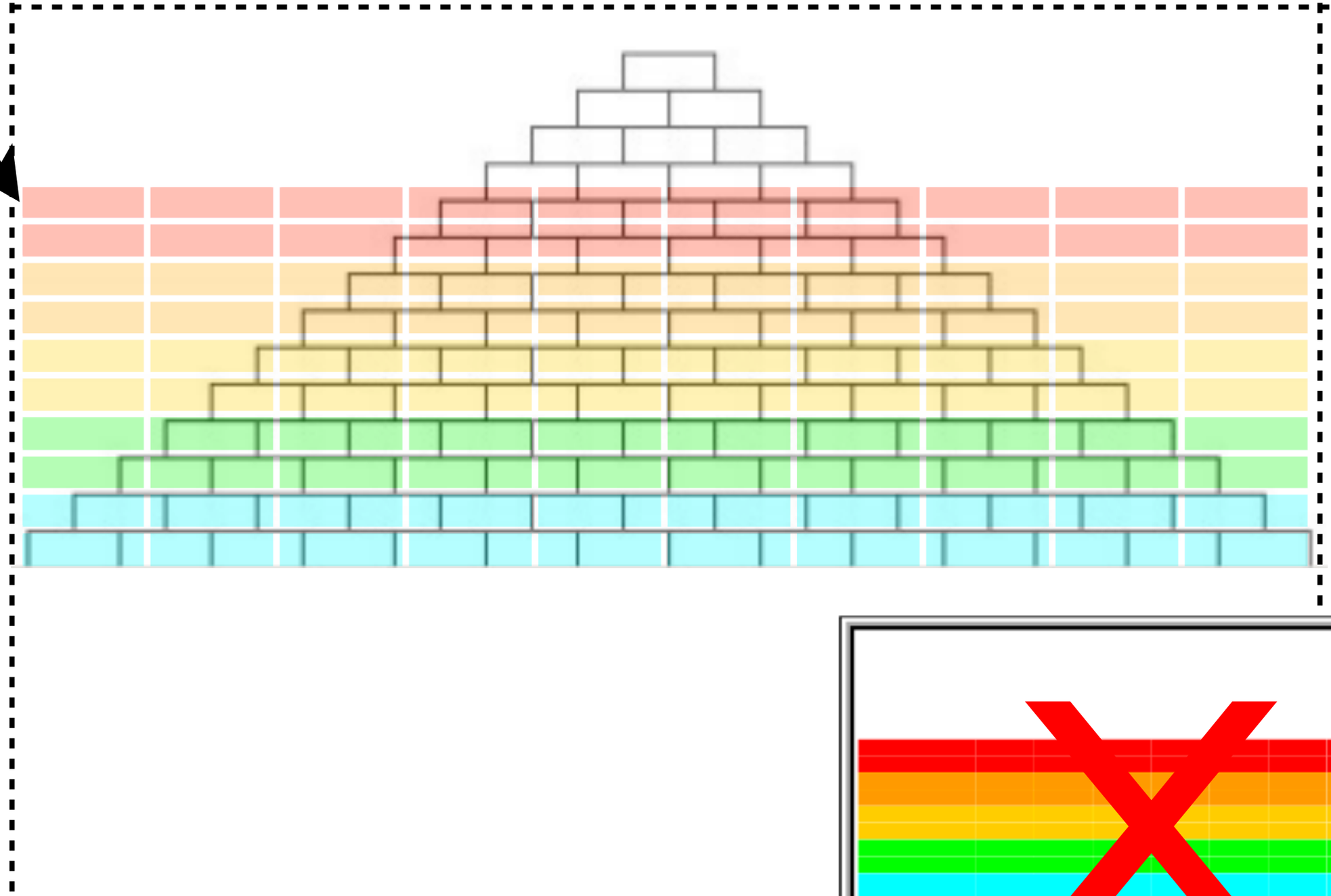
Breakout graphics



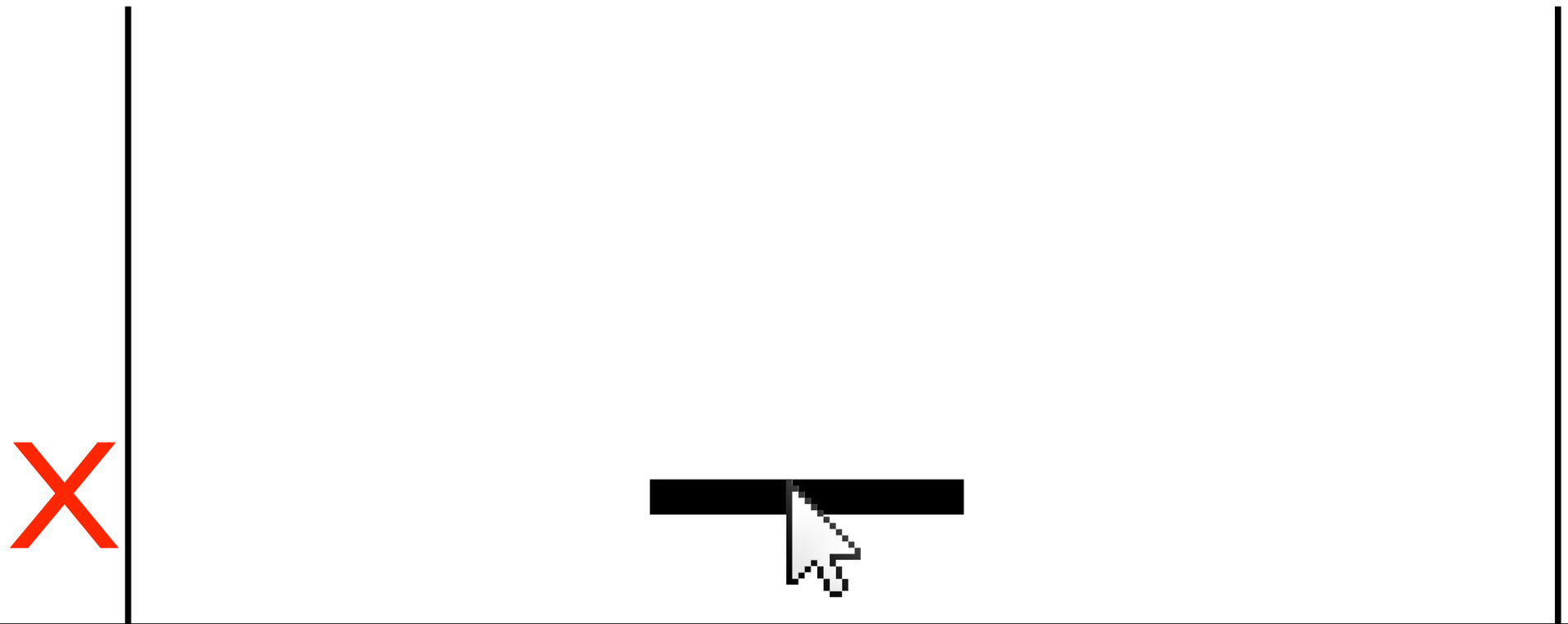
Constants

```
/** Width and height of application window in pixels */
public static final int APPLICATION_WIDTH = 400;
public static final int APPLICATION_HEIGHT = 600;

/** Dimensions of game board (usually the same) */
private static final int WIDTH = APPLICATION_WIDTH;
private static final int HEIGHT = APPLICATION_HEIGHT;

/** Dimensions of the paddle */
private static final int PADDLE_WIDTH = 60;
private static final int PADDLE_HEIGHT = 10;

/** Offset of the paddle up from the bottom */
private static final int PADDLE_Y_OFFSET = 30;

/** Number of bricks per row */
private static final int NBRICKS_PER_ROW = 10;

/** Number of rows of bricks */
private static final int NBRICK_ROWS = 10;

/** Separation between bricks */
private static final int BRICK_SEP = 4;

/** Width of a brick */
private static final int BRICK_WIDTH =
  (WIDTH - (NBRICKS_PER_ROW - 1) * BRICK_SEP) / NBRICKS_PER_ROW;

/** Height of a brick */
private static final int BRICK_HEIGHT = 8;

/** Radius of the ball in pixels */
private static final int BALL_RADIUS = 10;

/** Offset of the top brick row from the top */
private static final int BRICK_Y_OFFSET = 70;

/** Number of turns */
private static final int NTURNS = 3;
```

Bricks



Paddle

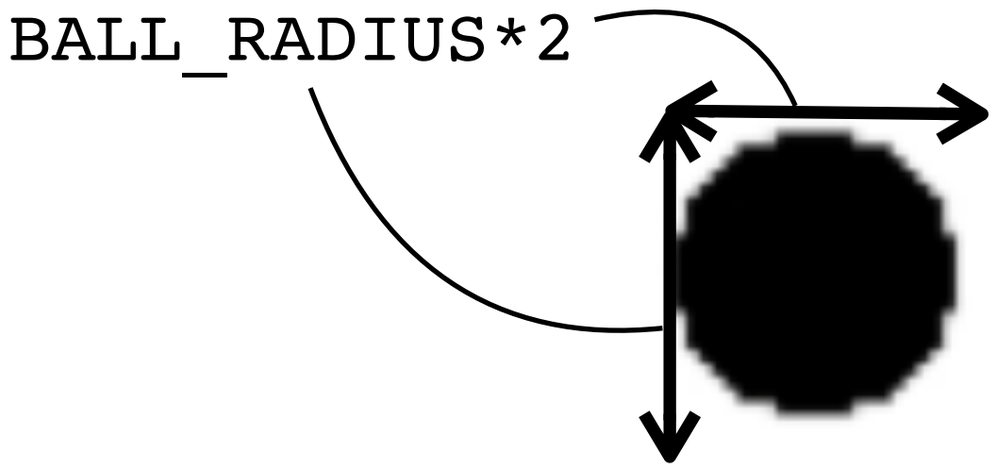


Mouse movement

`addMouseListeners()`

```
public void mouseMoved(MouseEvent e) {
  double mouseX = e.getX();
  double mouseY = e.getY();
  // ...
}
```

The ball



Which dimensions do the GOval constructor take?

Animation

```
while (not-done-condition) {
    update graphics  obj.move(dx, dy);
    pause(pause-time);
}
```

Ball movement

```
double vx;
double vy;
```

```
while (not-done-condition) {
    ball.move(vx, vy);
    pause(pause-time);
}
```

```
private RandomGenerator rgen = RandomGenerator.getInstance();
vx = rgen.nextDouble(1.0, 3.0);
if (rgen.nextBoolean(0.5)) vx = -vx;
```

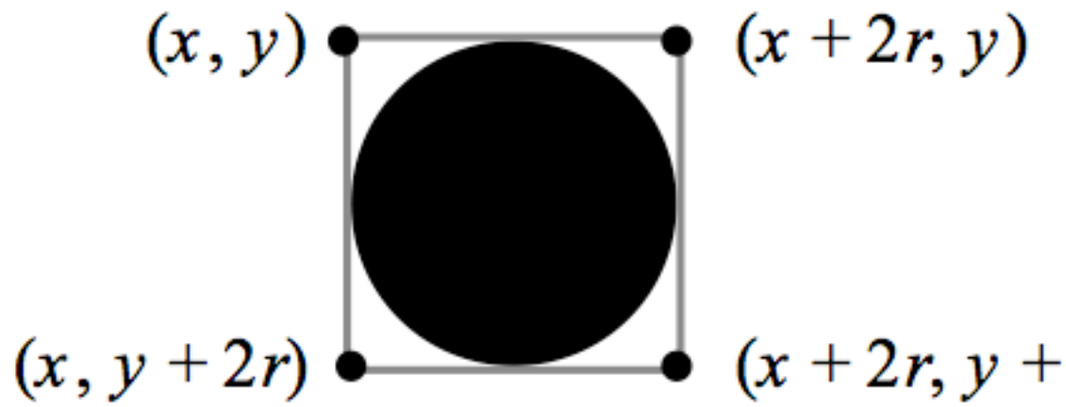
what's wrong here?

```
waitForClick();
```

~~nextDouble(-3.0, +3.0)~~

Collisions

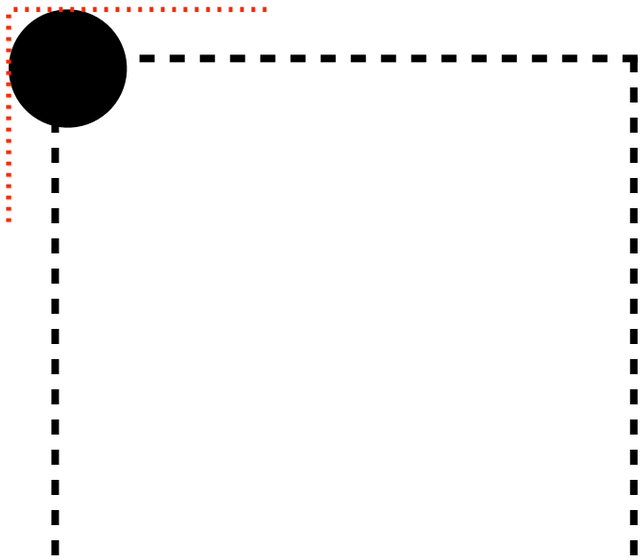
```
public GObject getElementAt(double x, double y)
```

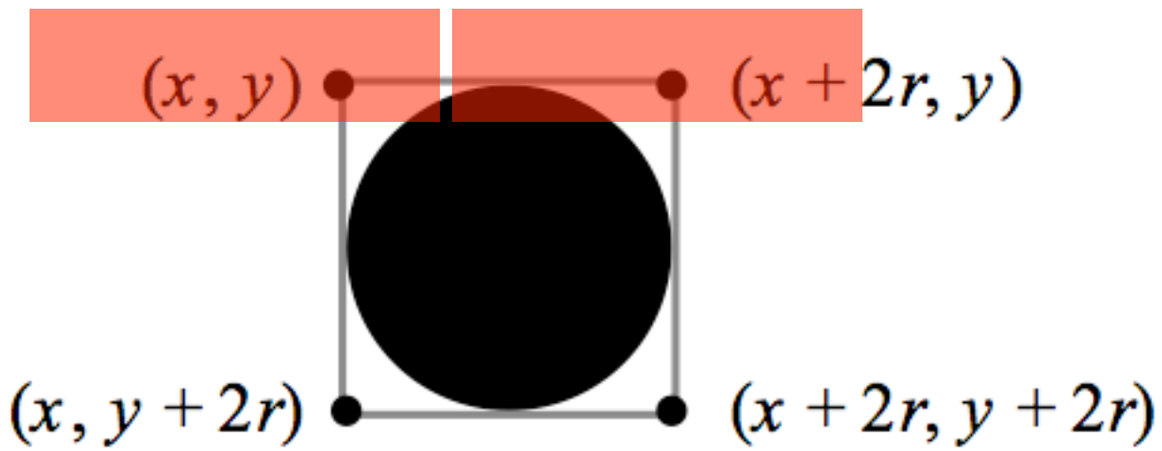


- Why not the middle of each side?
- Two types of collisions
  - A hit: `getElementAt != null`
  - A "hit": off the screen

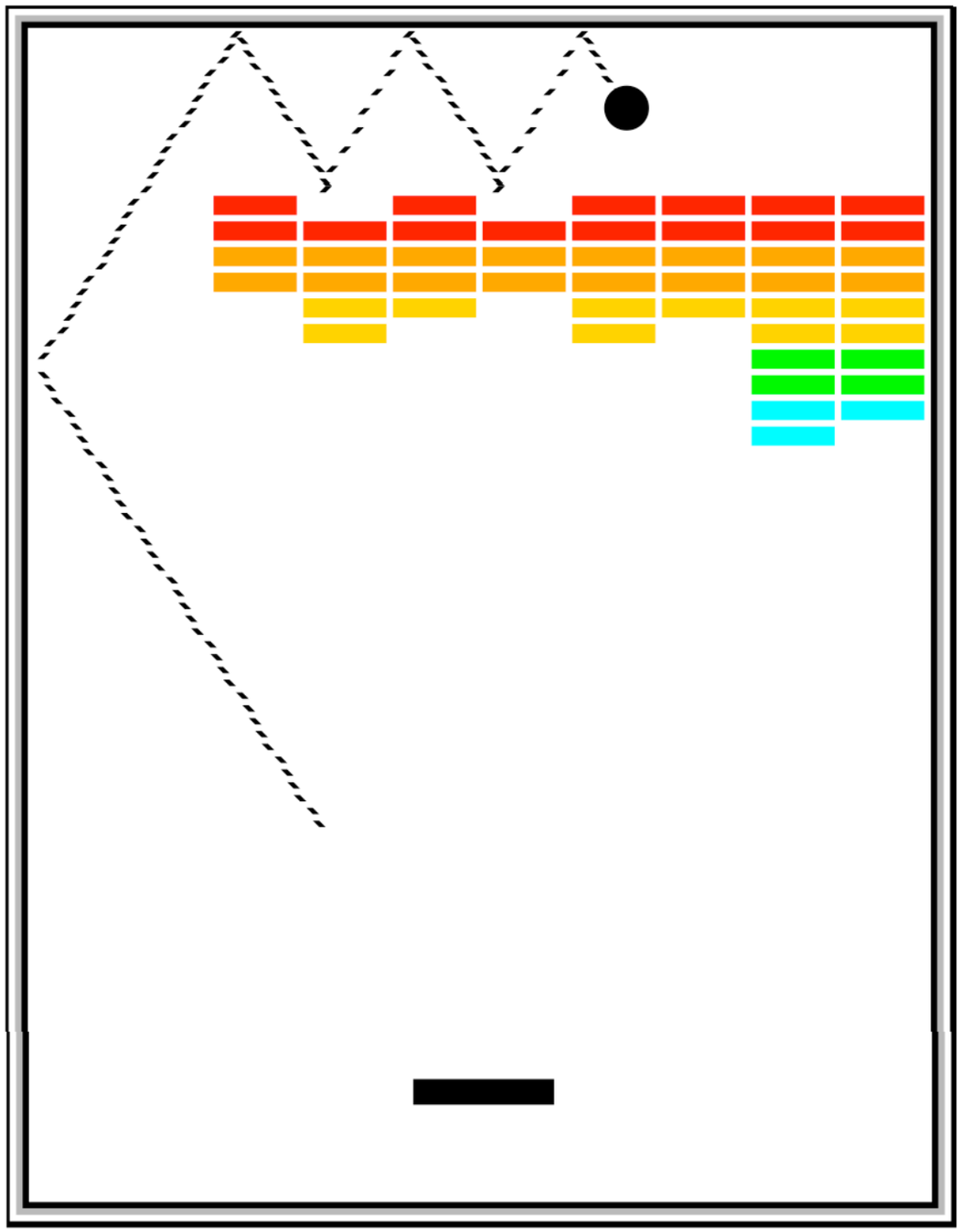
To consider...

- What gets inverted in a normal bounce?
- What if the ball "hits" multiple corners?





Playing the game



Ending the game

- Remove the ball once it goes off the screen, doesn't disappear automatically
  - `remove();`
- Detect winning and losing
  - how?
  - track bricks remaining

Testing

```
/** Width and height of application window in pixels */
public static final int APPLICATION_WIDTH = 400;
public static final int APPLICATION_HEIGHT = 600;

/** Dimensions of game board (usually the same) */
private static final int WIDTH = APPLICATION_WIDTH;
private static final int HEIGHT = APPLICATION_HEIGHT;

/** Dimensions of the paddle */
private static final int PADDLE_WIDTH = 60;
private static final int PADDLE_HEIGHT = 10;

/** Offset of the paddle up from the bottom */
private static final int PADDLE_Y_OFFSET = 30;

/** Number of bricks per row */
private static final int NBRICKS_PER_ROW = 10;

/** Number of rows of bricks */
private static final int NBRICK_ROWS = 10;

/** Separation between bricks */
private static final int BRICK_SEP = 4;

/** Width of a brick */
private static final int BRICK_WIDTH =
  (WIDTH - (NBRICKS_PER_ROW - 1) * BRICK_SEP) / NBRICKS_PER_ROW;

/** Height of a brick */
private static final int BRICK_HEIGHT = 8;

/** Radius of the ball in pixels */
private static final int BALL_RADIUS = 10;

/** Offset of the top brick row from the top */
private static final int BRICK_Y_OFFSET = 70;

/** Number of turns */
private static final int NTURNS = 3;
```

paddle

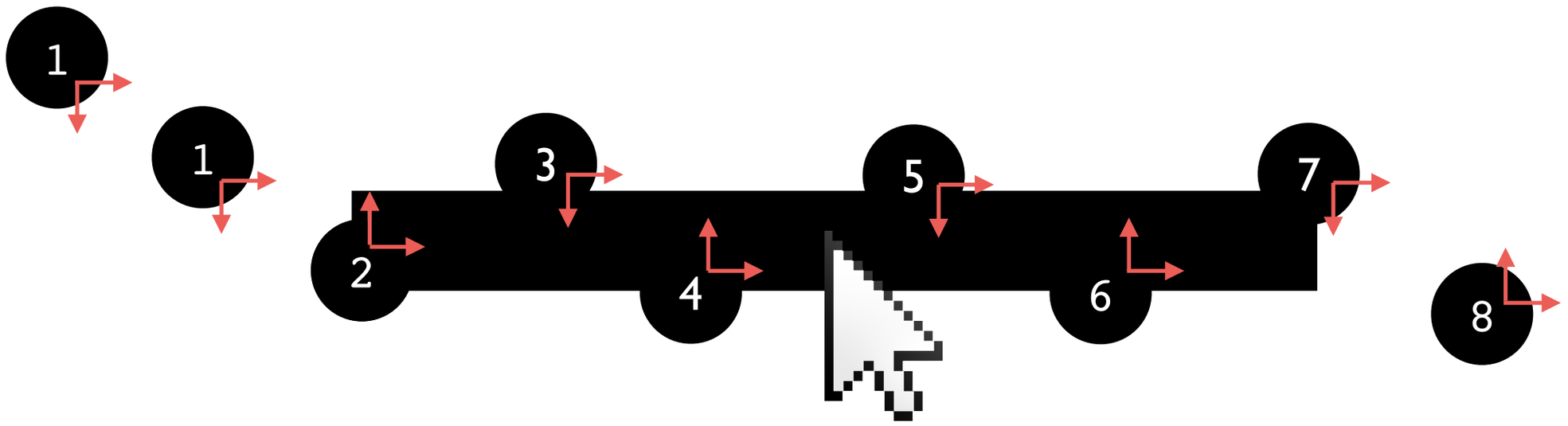
bricks

ball

gameplay

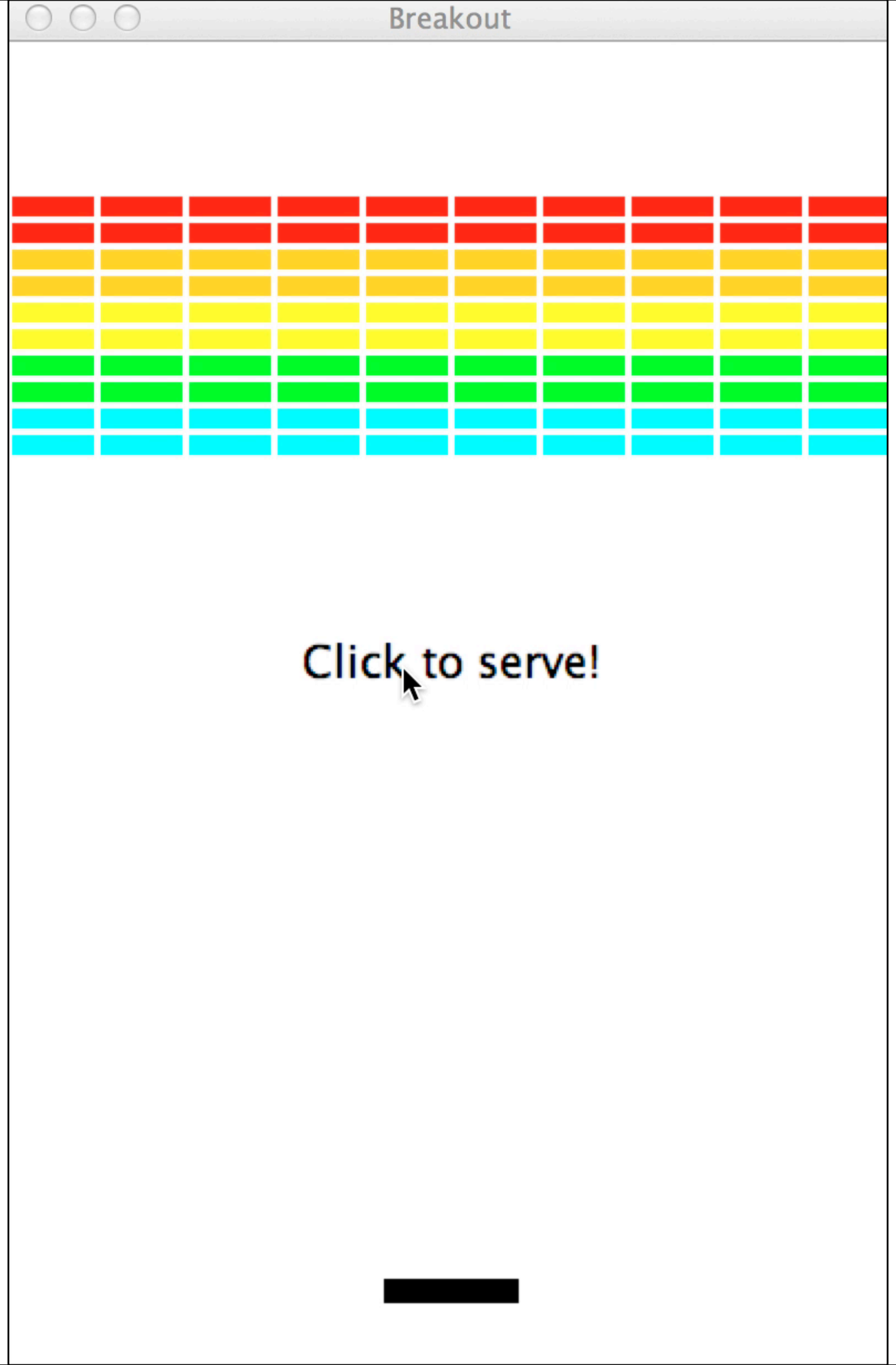
Try changing the boxed constants.
They each change an aspect of the game.
The game must still work.

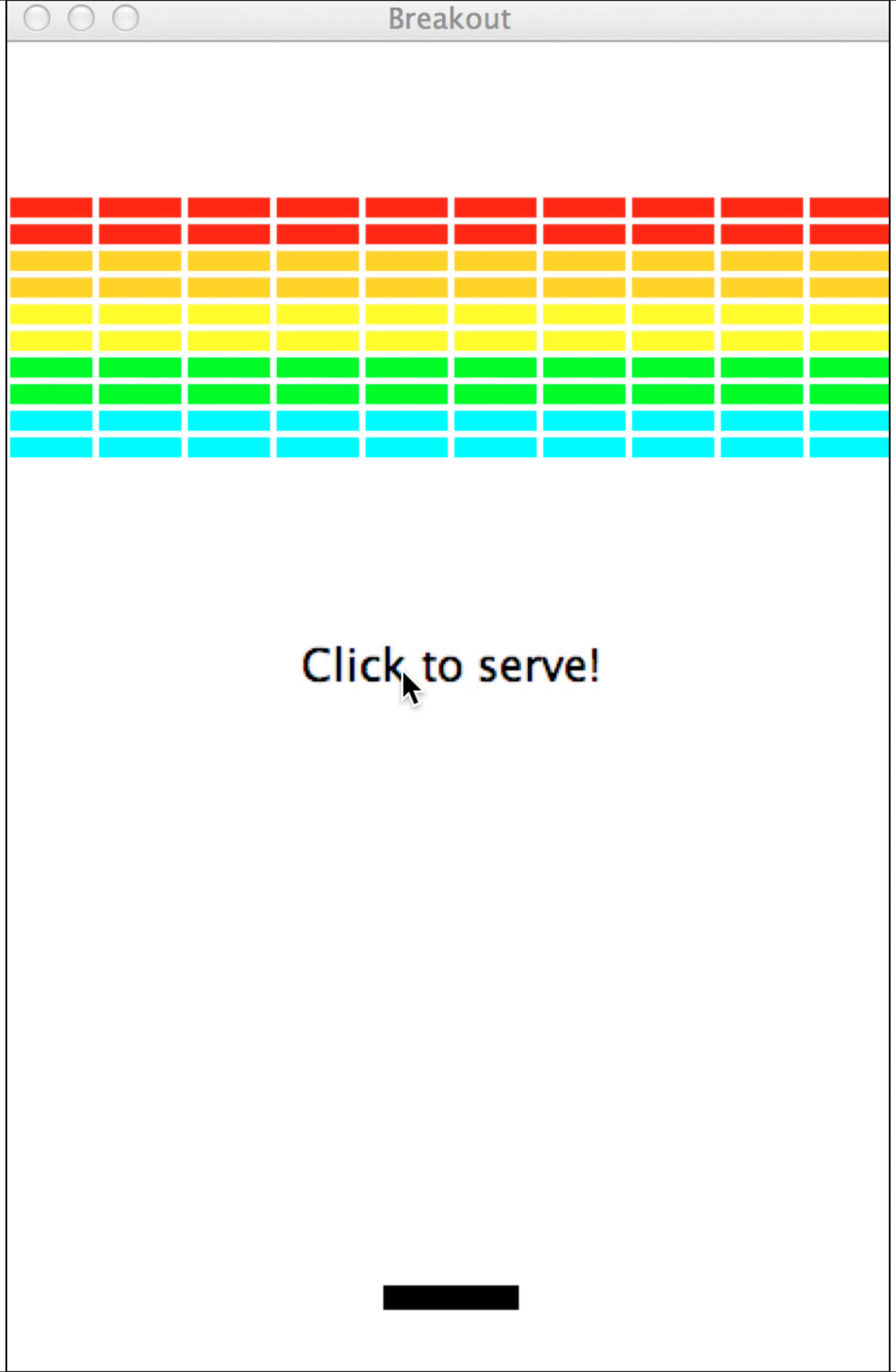
common bug:
ball stuck in paddle

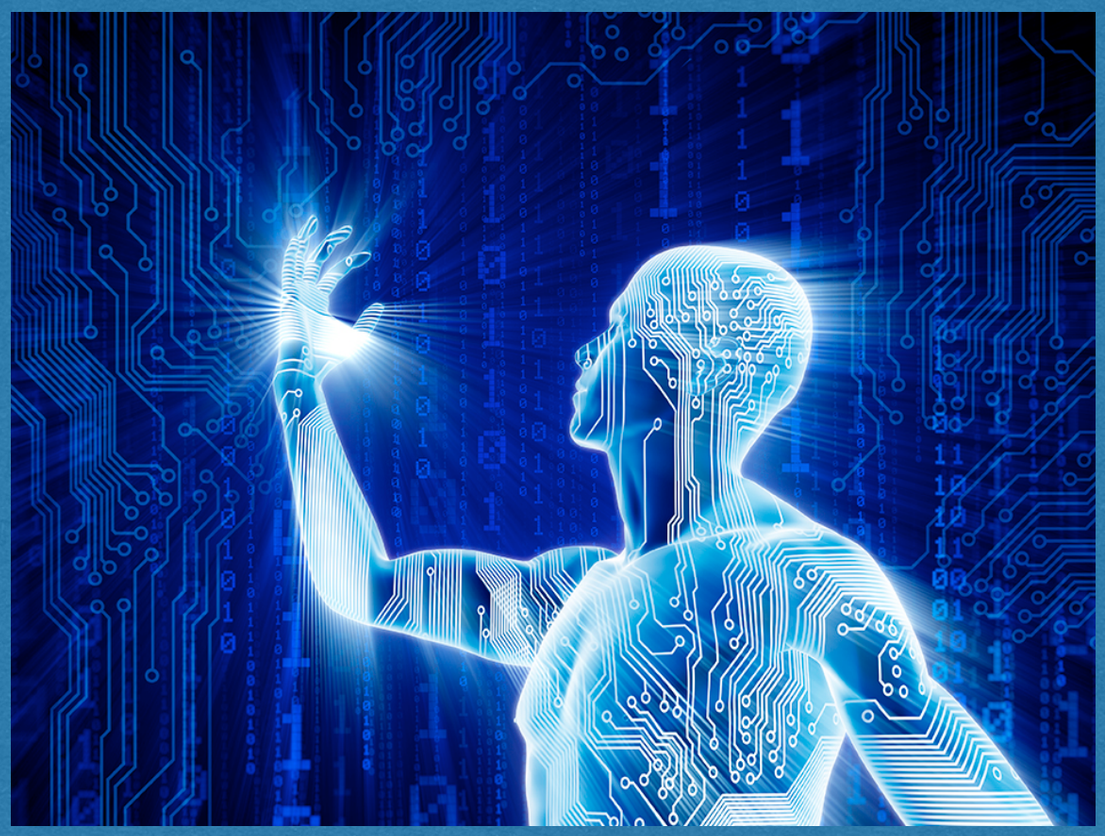


the auto-play trick!

a.k.a the one-line A.I.


Breakout
Click to serve!

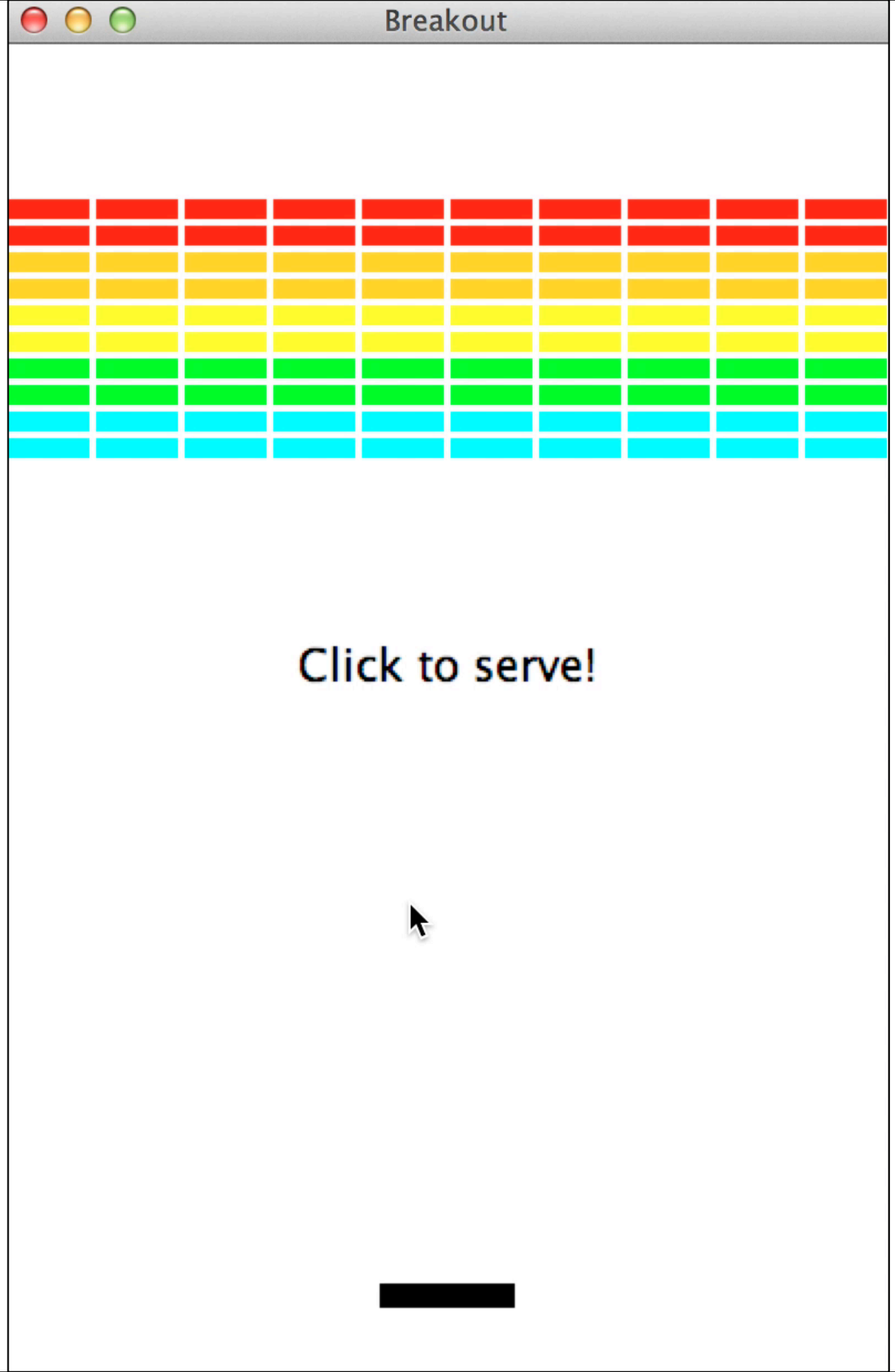




(+ one line of code)

extreme

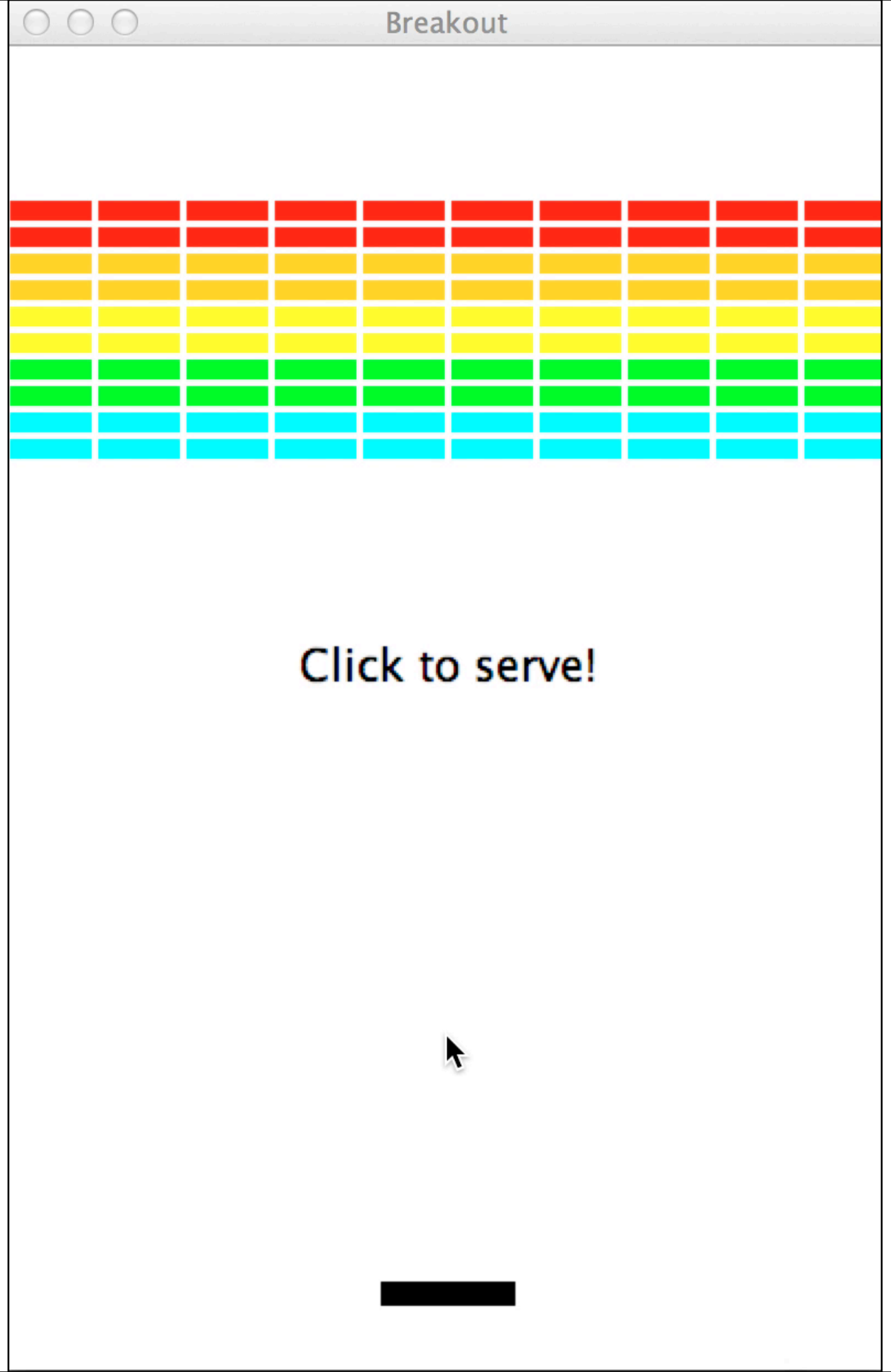
mode



SUPER

extreme

mode

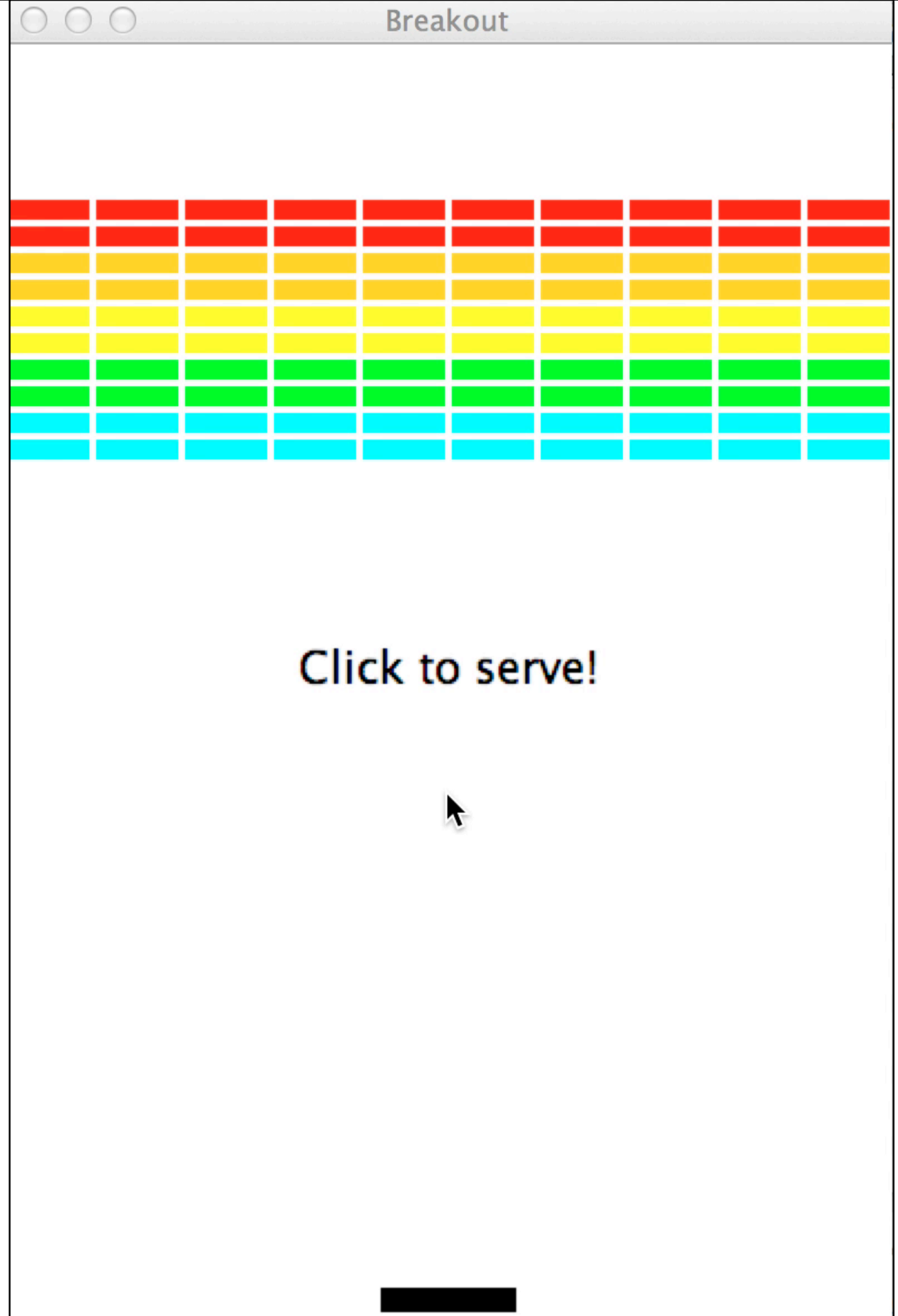


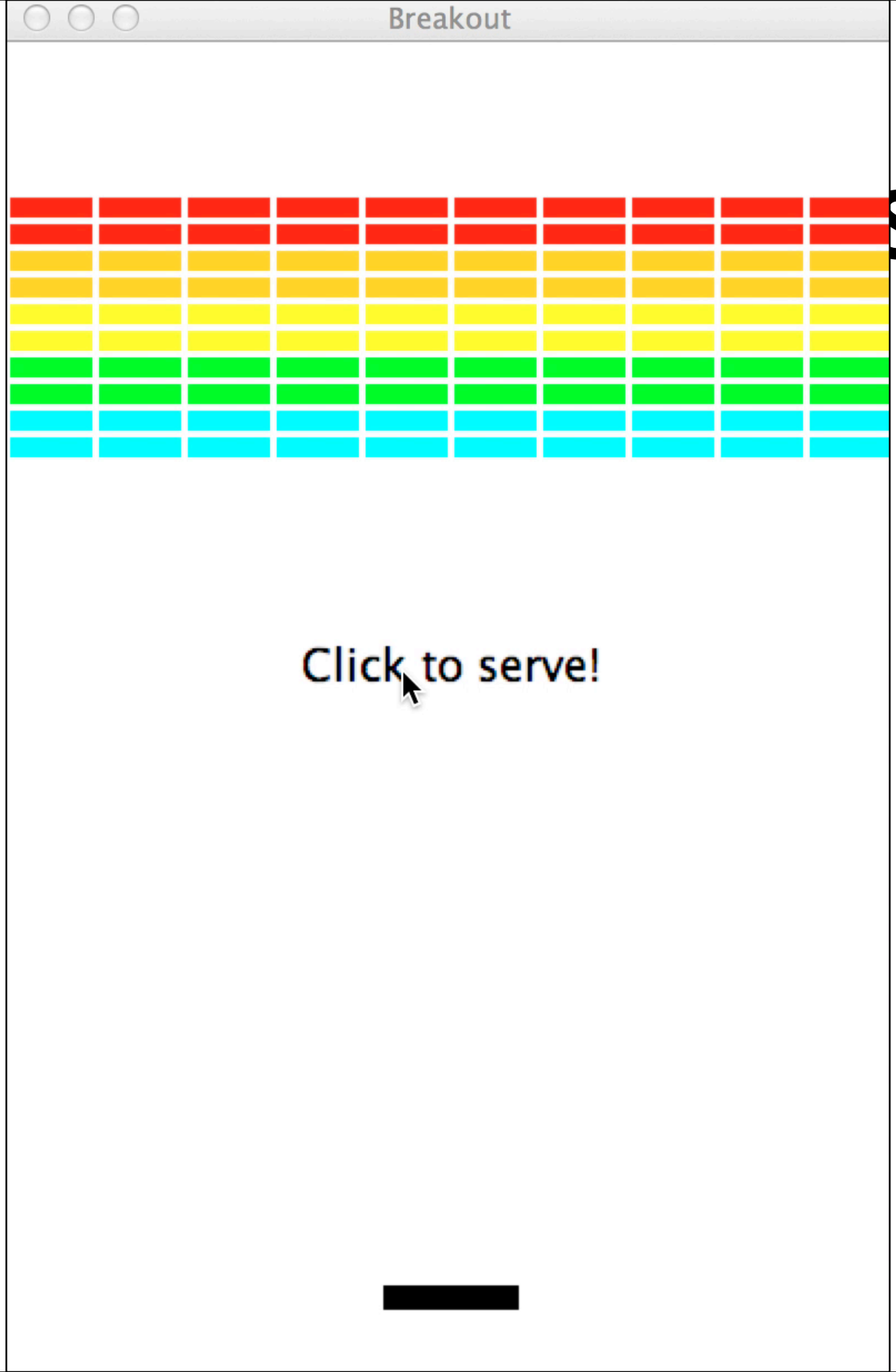
SUPER

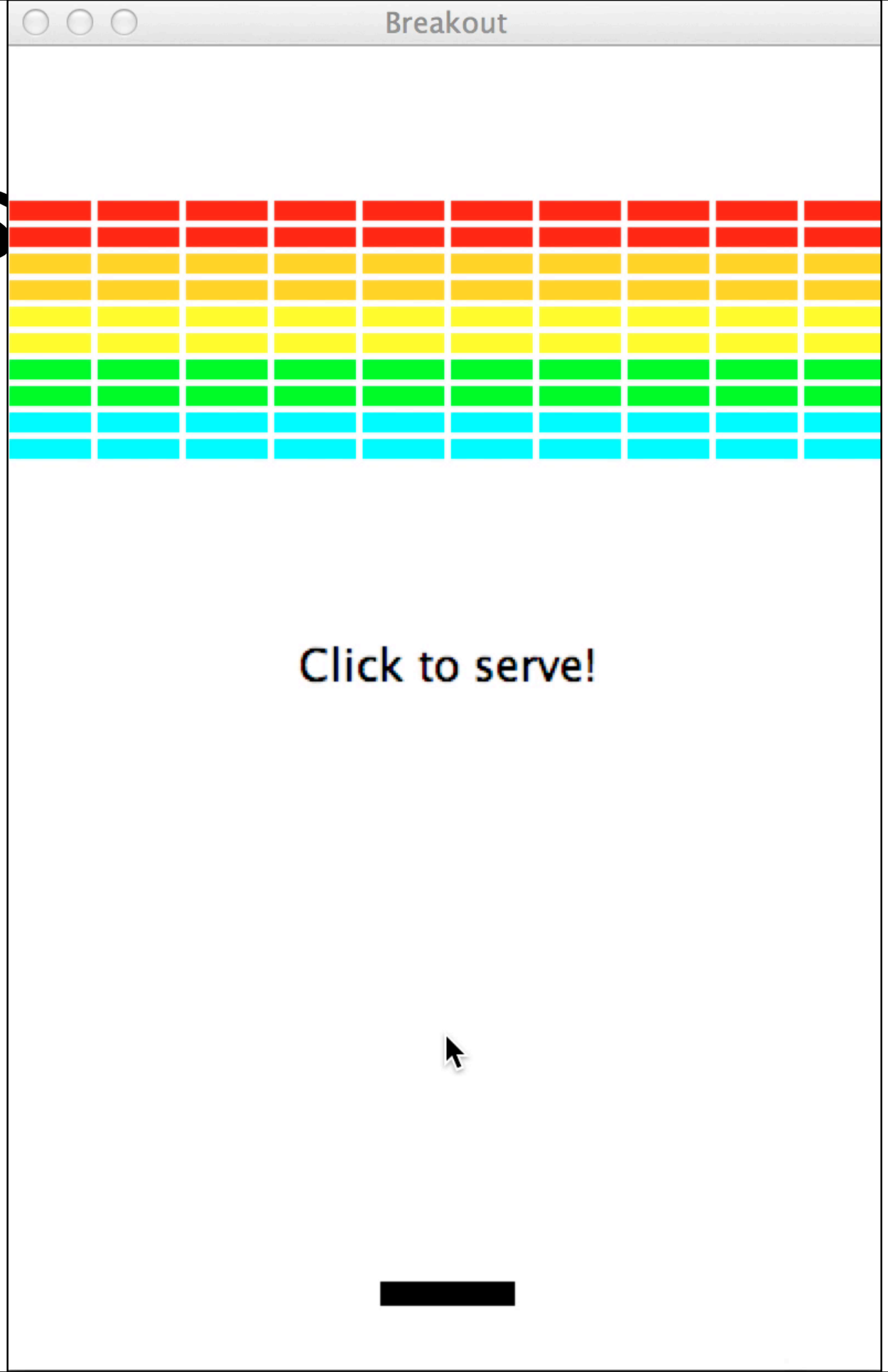
extreme

mode

MEGA




Breakout
Click to serve!


Breakout
Click to serve!

Style

Should I use an instance variable?

<u>General rules for when an instance variable is appropriate:</u>

1. If you need to access the variable in `mouseListener` methods, or
2. You access and change the variable ALL over the place, or
3. There's just no other way.

Avoid using instance variables unless you need them. It is poor style to make something an instance variable when it could have been a local variable.

Instance variables in Breakout

You'll justify each of your instance variables in the writeup.

- Ball? Yes, probably
- Bricks? No
- Paddle? Yes, definitely

Extensions

Breakout has a ton of possible extensions!

```
AudioClip bounceClip =
  MediaTools.loadAudioClip("bounce.au");

bounceClip.play();
```

- Follow the specifications carefully
- Comment
- Go to the LaIR if you get stuck
- **Incorporate IG feedback!**
- Have fun!