Timer-Based Animation

Jerry Cain
CS 106AX
October 6, 2023
slides leveraged from those written by Eric Roberts

Timer Events

- The programs we discussed last time respond to mouse events by installing event listeners into the **`GWindow`** object.
- JavaScript *also* allows you to listen for ***timer events***, which occur after a specified time interval.
- As with mouse events, you install a listener for a timer event in the form of a ***callback function*** that is automatically invoked at the end of the time interval.
- You can add animation to a JavaScript program by setting a timer for a short interval and having the callback function make small updates to the **`GObject`**s in the window.
- If the time interval is short enough (typically between 20 and 30 milliseconds), the animation will appear smooth to the human eye.

Timeouts

- JavaScript supports two kinds of timers. A ***one-shot timer*** invokes its callback function once after a specified delay. You create a one-shot timer by calling

```
setTimeout(function, delay);
```

  where *function* is the callback function and *delay* is the time interval in milliseconds.

- An ***interval timer*** invokes its callback function repeatedly at regular intervals. You create an interval timer by calling

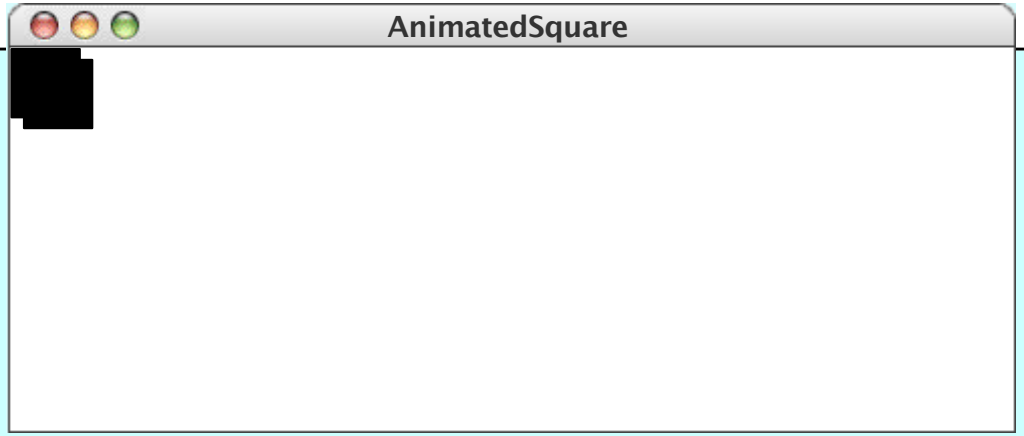
```
setInterval(function, delay);
```

  The `setInterval` function returns a numeric value that you can later use to stop the timer by calling `clearInterval` with that numeric value as an argument.

A Simple Example of Animation

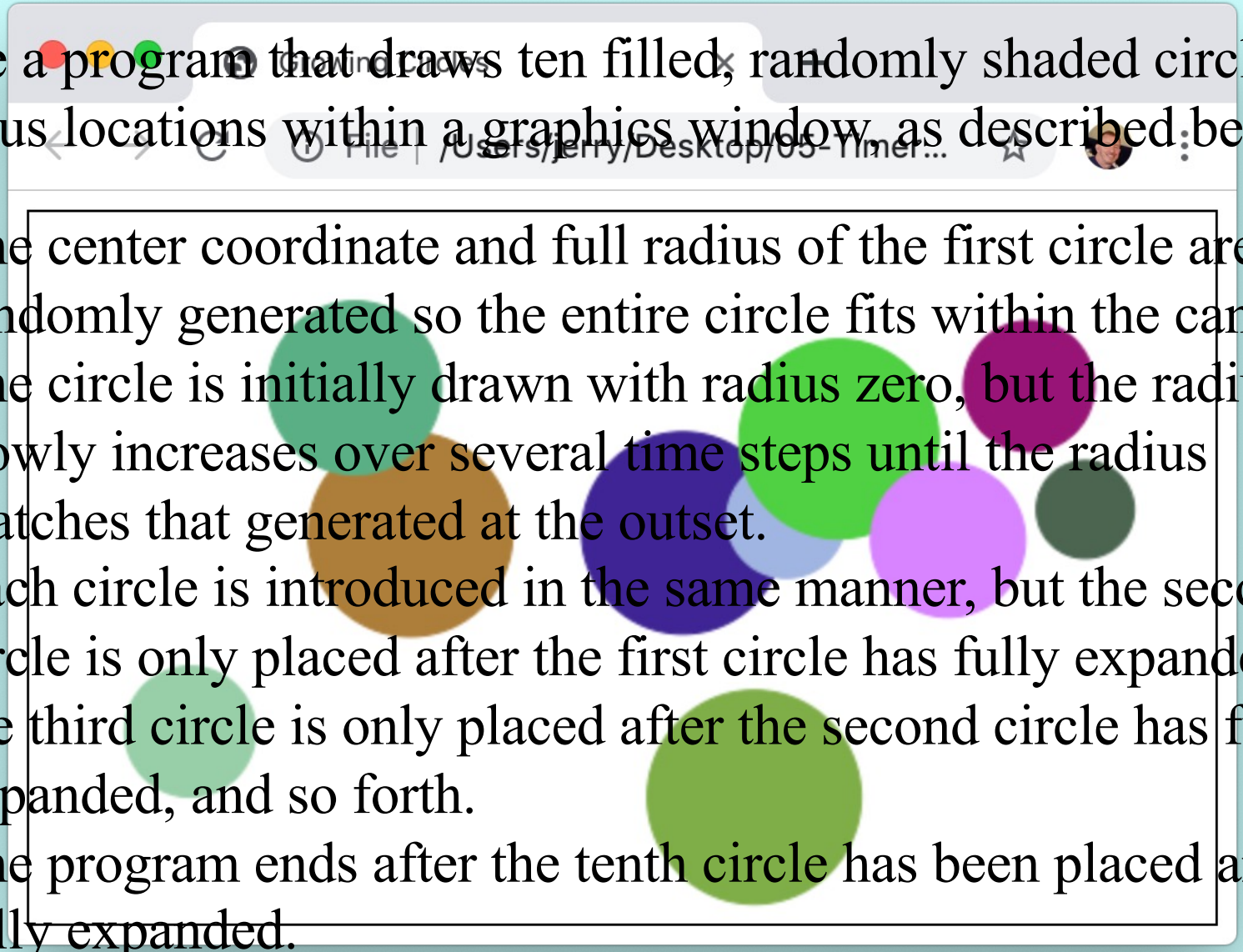
```
function AnimatedSquare() {
  function step() {
    square.move(dx, dy);
    stepCount++;
    if (stepCount === N_STEPS) clearInterval(timer);
  }
```

gw	dx	dy	square	stepCount	step	timer
	4	2	■	1	...	1729



Exercise: Exploding Circles

Write a program that draws ten filled, randomly shaded circles at various locations within a graphics window, as described below:



- The center coordinate and full radius of the first circle are randomly generated so the entire circle fits within the canvas.
- The circle is initially drawn with radius zero, but the radius slowly increases over several time steps until the radius matches that generated at the outset.
- Each circle is introduced in the same manner, but the second circle is only placed after the first circle has fully expanded, the third circle is only placed after the second circle has fully expanded, and so forth.
- The program ends after the tenth circle has been placed and fully expanded.

The End