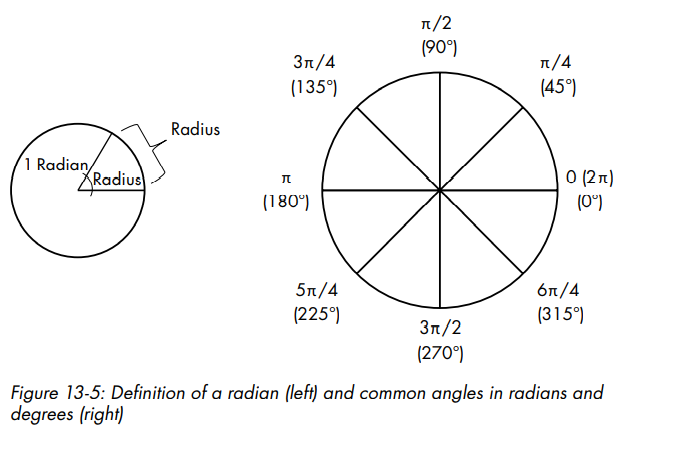
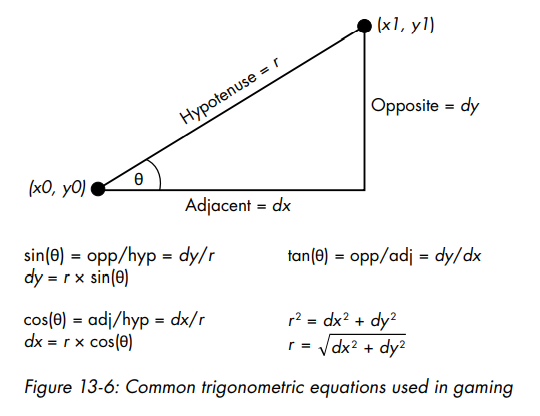
A radian is a standard unit of angular measurement equal to the angle made when the radius is wrapped around a circle.

One radian is slightly less than 57.3 degrees.

To convert degrees to radians, you can either multiply the degrees by π and divide by 180—like a chump—or just use the math module!



Objects move in pygame by increments of x and y. The direction and speed of a particle are used to get its delta-x (dx) and delta-y (dy) vector components. These represent the difference between a particle’s initial position and its position after completion of a single game loop. You calculate vector components using trigonometry. Useful trigonometric equations are provided in Figure 13-6.



For the angle ϴ, you use the orient variable. The self.vel attribute equates to r. Knowing these two components, you can use a trigonometric equation to derive self.dx and self.dy ξ. To derive self.dx, multiply self.vel by the cosine of orient, and to derive self.dy, multiply self.vel by the sine of orient. Note that you must make self.dy negative, as the particles are ejected upward and y-values in pygame increase downward.

To render the game objects and update the visual display, pygame uses a process called blitting. Blit stands for block transfer, which is just copying pixels from one rectangular Surface object onto another. By blitting the background onto the screen, you cover the screen with the Io image. With blitting, you can take the same image and draw it multiple times at different places on the screen. It can be a slow process, so game developers use clever techniques to address this deficiency, such as blitting only around areas that are currently being updated, rather than blitting the whole screen in every game loop.

Flipping is a type of double buffering where you blit everything from the screen object to the actual display. Flipping gets around the inherently slow process of displaying graphics, which can cause the screen to flicker, by doing the work on a behind-the-scenes rectangle and then using a version of the blit() method to copy to the final display