

## Lesson Two: Gravity and Wind

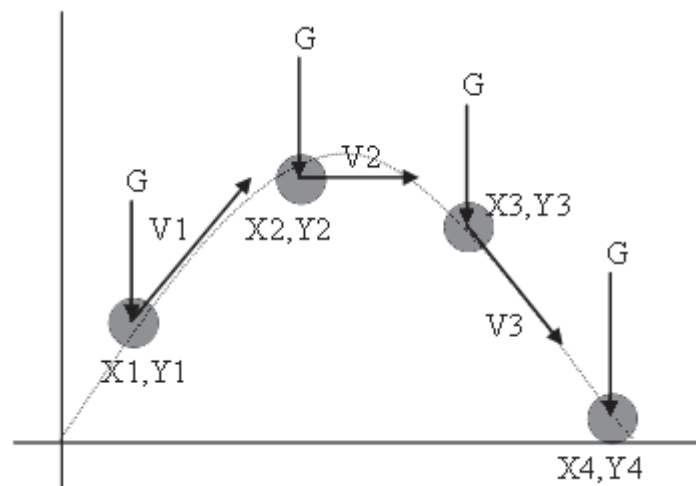
Many computer games involve the use of objects that are either thrown or shot into the air. The movement of these objects may be influenced by gravity and wind effects.

### Gravity

In the real world, every object that is launched or thrown into the air is influenced by the force of gravity. The Earth's gravity is a force which is constantly pulling everything towards the center of the earth (or, in terms of the computer screen, straight down in the positive Y direction).

When you throw a ball into the air it will travel up and away from you. As soon as it leaves your hands, gravity begins to work on the ball. Depending on how hard you throw, the ball will eventually stop going in the upwards direction and will start moving down towards the ground. The arc that the ball follows from the time you throw it until the time it hits the ground is called a parabolic arc, which is shaped like an upside-down 'U'.

Let's take a look at a diagram that shows a ball moving under the force of 'gravity' on the screen:



The original position of the ball is shown as point  $(X_1, Y_1)$ . The ball has an initial velocity of  $V_1$ , which is moving the ball in the negative Y direction (up), and positive X direction (to the right). We also have the force of gravity 'G', which is pushing down on the ball. The effect is to reduce the upward velocity in the Y direction and leave the X velocity untouched.

The next position, point  $(X_2, Y_2)$ , shows where the ball is after one or more iterations of the `Update()` method. At this point, gravity has slowed the Y velocity to near zero, and our ball is mostly moving in the positive X direction (still to the right).

The next position, point  $(X_3, Y_3)$  again shows the position of the ball after another timer tick or two. Now, gravity has reversed the Y velocity so that the direction is now positive (moving down on the screen). Finally, the last position of the ball  $(X_4, Y_4)$  shows the ball where it has landed on the "ground".

The dotted line in the diagram shows the parabolic arc the ball would complete in the real world. Notice that the speed of the ball from left to right in the X direction is completely unaffected by the force of gravity. Gravity only acts to change the velocity in the Y direction. Also notice that the force of gravity “G” is a constant that affects the ball equally at each position.

We now know the values for both the X and Y acceleration components! There is zero (0.0) acceleration in the X direction because the X velocity does not change. The Y acceleration is a constant “G” representing the force of gravity. The second version of `Sprite.Accelerate()` accepting the component AX and AY values is ideal for our use in this situation.

So, to add the effects of gravity to a sprite flying through the air, we could use the following code:

```
mySprite.Accelerate(0.0f, 0.04f);
```

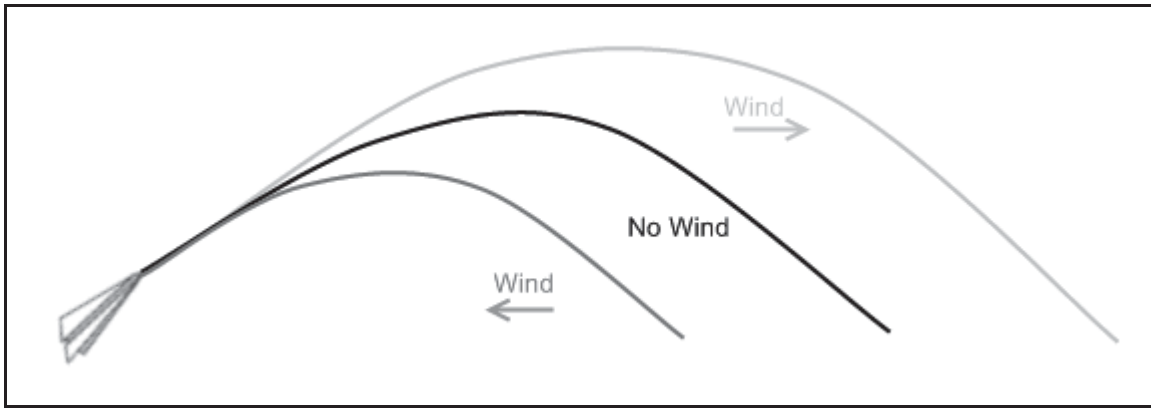
While gravity may be known as  $9.8 \text{ m/s}^2$  in the real world, we don’t have the concept of a “meter” on the computer screen. We are also applying this force 60 times a second (assuming default timing for the `Update()` function call). You’ll have to play with several different AY values to find the strength of gravity value that “feels” right for your game environment. The size of the AY parameter will determine how quickly the object will fall back down to the ground.

## Wind

Up until now we have only considered projectiles that are travelling through the air without any impact from the wind. Wind affects almost every projectile in the air. Golfers often check the wind before lining up their shots. Paper airplanes fly further if they are propelled by the wind, and baseball players may hit more home runs with the wind at their back than if the wind is blowing in their face. Adding the effects of wind for flying objects in a game may be a nice touch of realism or even be crucial to game-play!

So how do we add wind to a game? In the last section, we learned that gravity is a constant acceleration in the Y direction but did not affect the X velocity. The force of gravity caused our projectile to move in an arc on the screen. Wind, on the other hand, can be considered a constant acceleration in the X direction (assuming it is simply blowing left or right with no up or down-drafts). A wind blowing to the left would be a negative X acceleration, and wind blowing to the right would be a positive X acceleration.

The wind can have a powerful affect on the velocity and trajectory of a projectile. Consider the following illustration of a paper airplane in flight:



The paper airplane is thrown three different times with the same initial starting location, direction, and speed. The black line represents the path that the airplane will take if there is no wind at all. With no wind, only the force of gravity affects the plane's path.

The light gray line shows the path the airplane will take if the wind is heading in the same direction as the airplane. This type of wind is called a *tailwind*, which is any wind that is travelling in the same direction as the projectile. The airplane will travel much further with a tailwind!

The dark gray line shows the path the airplane will take when the wind is travelling in the opposite direction of the airplane. This type of wind is called a *headwind*. With a headwind the airplane's flight will be much shorter!

How do you implement the effects of wind in your game? Easy! You already have an acceleration function that will split apart acceleration in either the **X** or **Y** direction. You added gravity to a projectile by specifying acceleration in the **Y** direction. The wind can just be thought of as acceleration in the **X** direction. A positive **X** acceleration will simulate a wind that travels from left to right on the screen. A negative **X** acceleration will simulate a wind that travels from right to left on the screen.

To add both gravity and wind values to a sprite that is flying through the air, we could use the following code:

```
mySprite.Accelerate(0.03f, 0.04f);
```

The value 0.03 will give the effect of wind blowing from left to right on the screen. To add a leftward wind instead, just use a negative value like -0.03. Again, just like gravity, you'll have to play with the magnitude of this number to see what feels right in your game. Pick a value that is too high and you've just created a tornado that will overwhelm every projectile motion in your game!