Lesson 4 (2 - 3 days)

Lesson Overview
Students will learn how to utilize sprits and develop them in AppInventor to build out a more complex game.

Objectives
Students will understand how sprites are constructed and manipulated through four different methods.

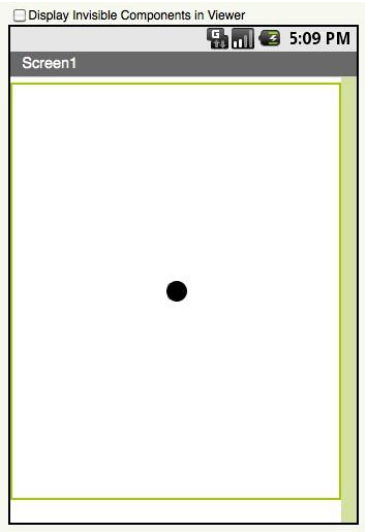
Activity
Introduce the **Ball** component as one of two Sprite components in AppInventor that have the ability to move around on a canvas. The second sprite component is **ImageSprite**, which is similar to a ball in that it can be moved around on a canvas, but it is visually represented by a custom Image that is uploaded by the AppInventor developer.

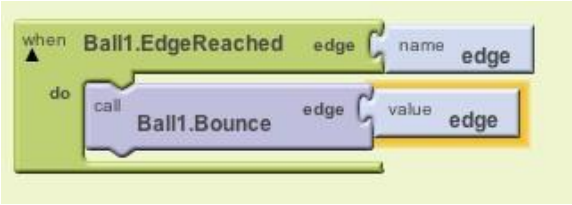
Explain that there are many ways to make Balls and ImageSprites move, but in this course we'll discuss the four of them (listed below) that are most applicable to game development. Students should make notes of the four types as they are going to develop each method.

1) Automatically (using built-in properties)
2) Using buttons
3) Using the AppInventor Clock component
4) Using Accelerometer

Motion Method 1: Automatically

Introduce sprite properties such as **Heading**, **Speed**, **X**, **Y**, and **Interval** that are required for automated motion of the sprite, and show students how to set these properties from the Properties panel in the Designer view. In order to make the ball move, we will update the ball's x position as the left and right buttons are pressed. The amount of pixels that we move the ball *each* time the buttons are clicked is known as the ball's *speed* (in the x direction). Create a simple app where the ball bounces around the screen, using the 'EdgeDetected' event handler and the 'Bounce' procedure in the blocks editor.

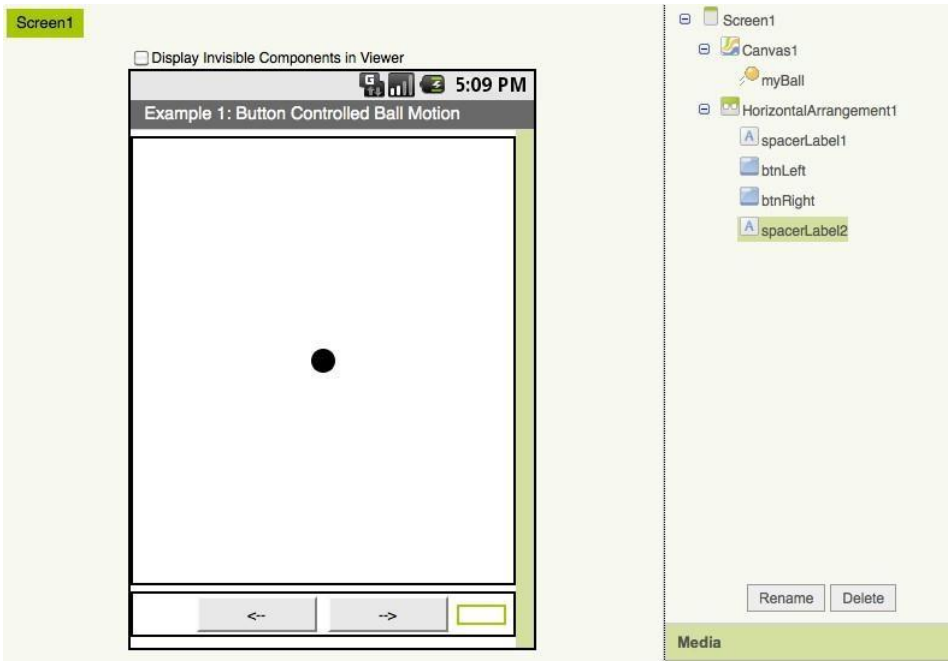




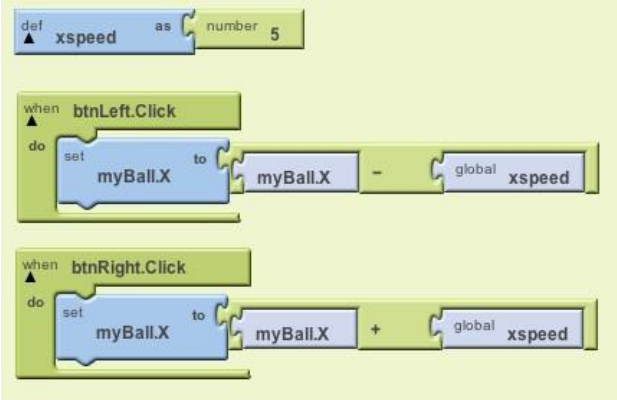
Then, ask the students to replace the ball with an **ImageSprite**, using an image of their choice but make it do the same sort of bouncing around using 'edge detected'. The ball should bounce off each edge if one correctly.

Motion Method 2: Button-controlled ball motion:

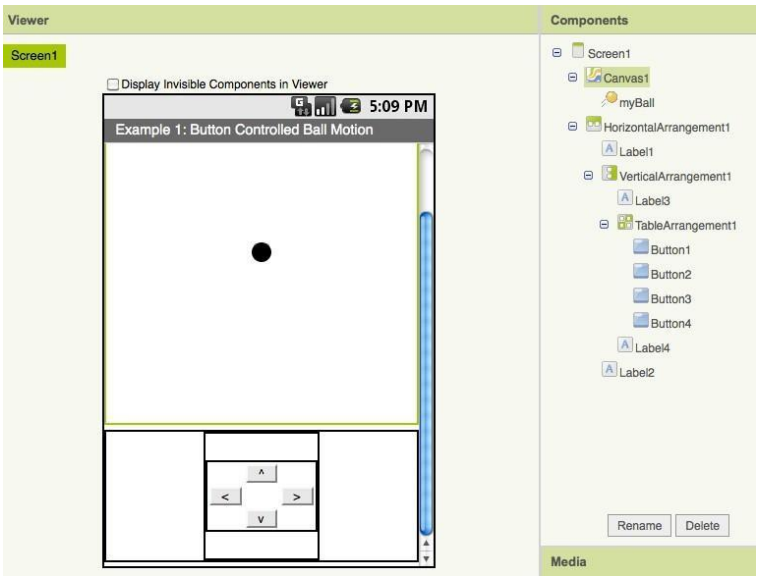
Ask students to create an AppInventor project with a single ball on a canvas and left/right buttons below the canvas, as shown below:

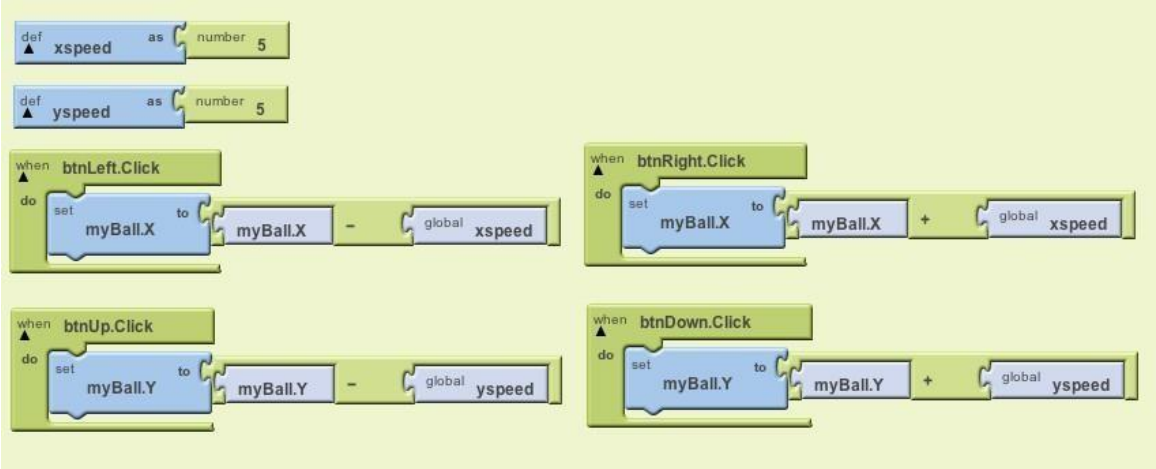


Introduce the concept of **variables**, and discuss how to define your own, and how to access and set their value. Then, ask students to create a variable representing the speed of the ball in the x direction. The blocks code for making the ball move then looks as follows:



Ask students to *extend* this app for up and down motion controlled by up and down arrows:

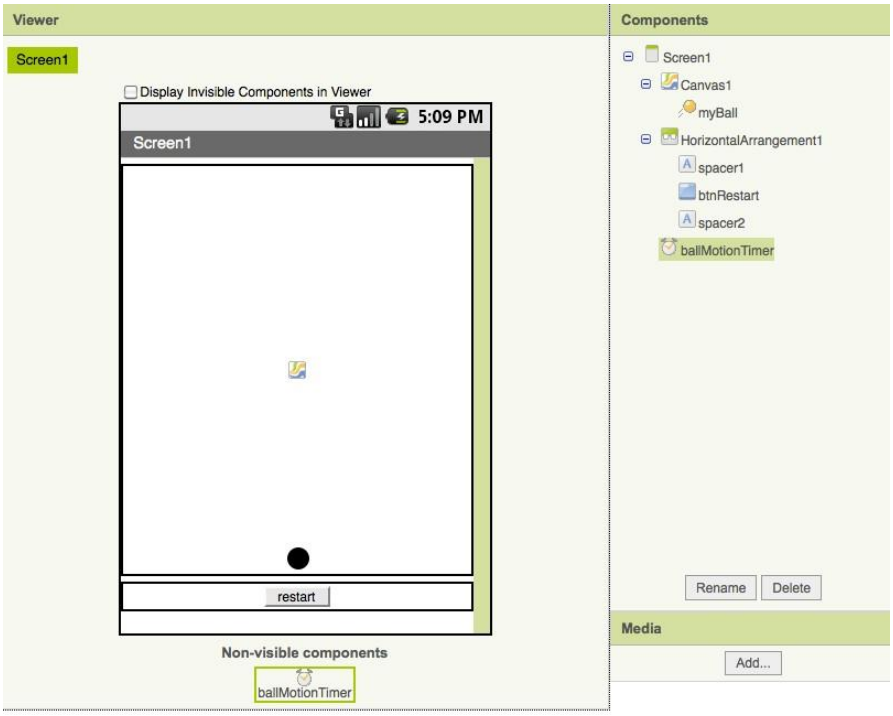




Motion Method 3: Using the AppInventor Clock component

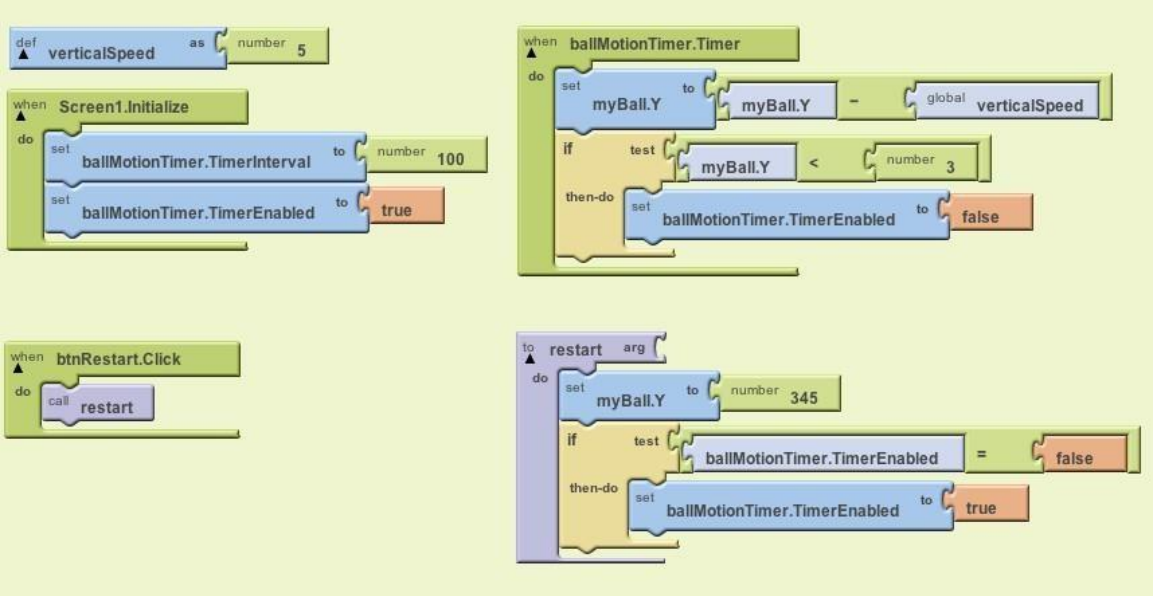
Introduce the **Clock** component, which is a timer that fires every few milliseconds (as specified by the **TimerInterval** property, which can be set either in the designer view or in the blocks editor).

Ask students to create a new project with a Ball, Canvas, and Clock (which will show up below the screen as a **non-visible component**, meaning it is not seen in the final app).



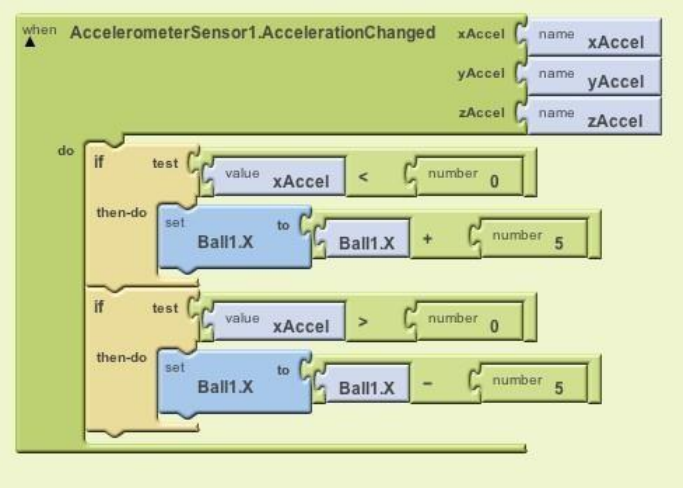
We now make the ball move up a few pixels (as determined by the value of the speed variable) every time the timer fires. Once the ball reaches the top of the screen, we ask the timer to stop firing, and we wait for the user to click the 'restart' button. Clicking the restart button causes the ball to reset its location to the bottom of the screen and turns on the timer once again.

We use this opportunity to teach how to define and call your own **procedure**, by turning the 'restart' action into its own procedure. The blocks code is shown below:

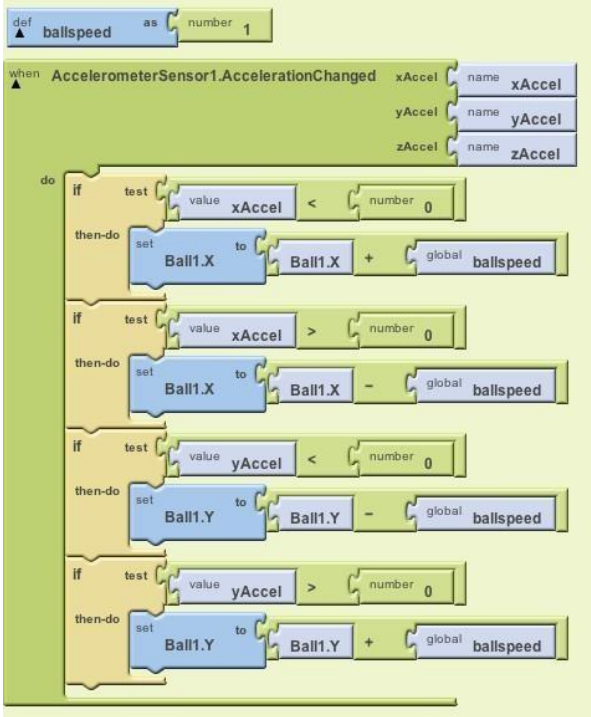


Motion Method 4: Using the Accelerometer

The Accelerometer is a built-in sensor in Android devices that senses changes in acceleration. Here, we use the accelerometer to make the ball move left and right as the phone is tilted left and right, hence avoiding the need for buttons and other forms of tactile input.

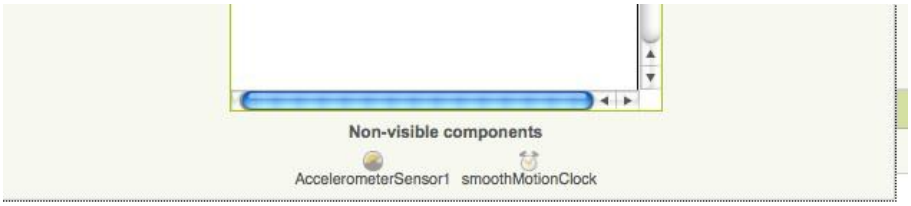


If students finish early, ask them to extend the program to also detect acceleration changes in the y direction (to move the ball up and down):

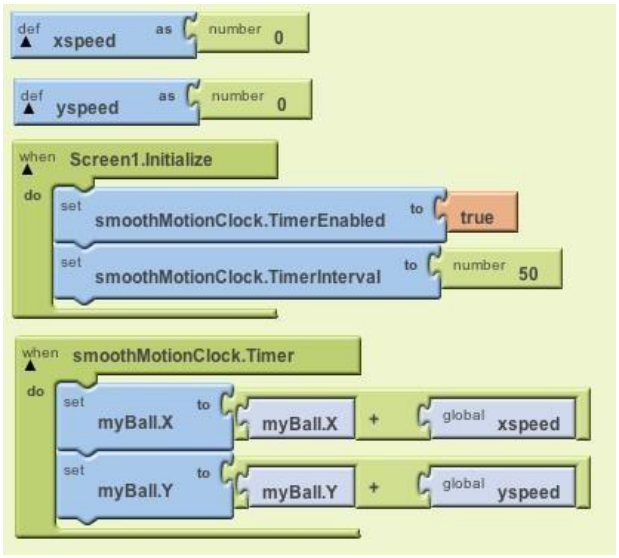


Extra Credit: Students will note that while the above code works, the motion of the ball is not smooth. In order to achieve smooth motion, students should use the **Clock** component *in conjunction* with the **Accelerometer** sensor, following the steps below.

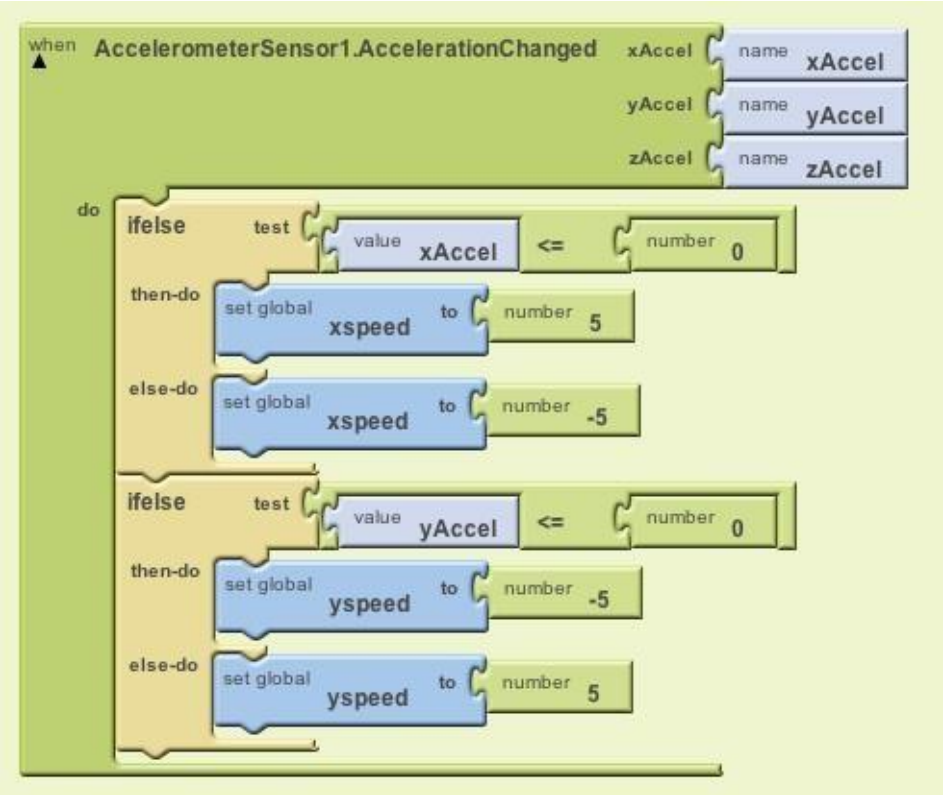
Smooth Motion Step 1: Add a Clock component to the project. In the below screenshots, the clock is called "smoothMotionClock".



Smooth Motion Step 2: In the blocks code, create variables for speed in the x and y direction, initialize the clock's timer interval in the Screen's Initialize event, and handle the clock's Timer event by updating the ball's x and y location by the appropriate speed variables.



Smooth Motion Step 3: Inside the AccelerometerSensor component's AccelerationChanged event handler, instead of updating the ball's position as you did in the non-smooth version of this application, simply set the value of the x and y speed variables. Now, the ball's position will be updated by the Clock timer and will be unaffected by factors such as how quickly the phone is tilted, resulting in regular and smooth motion.



#Code{RED}