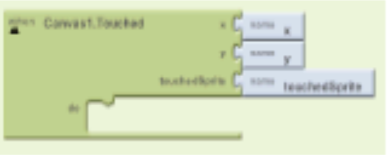
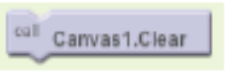
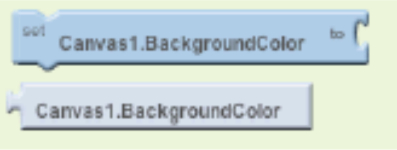
AppInventor Flash Cards



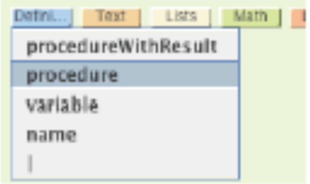
Events are actions that can be detected and responded to.



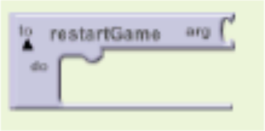
Procedures are operations that can be performed by components, e.g. "Clear" is an operation on Canvas objects.



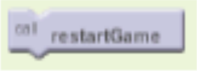
Properties are qualities of an object that can be read and modified.



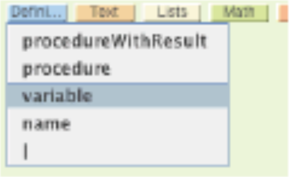
Click anywhere on the screen, and a colorful menu should pop up. On the "Define..." tab, click "procedure" or "procedureWithResult".



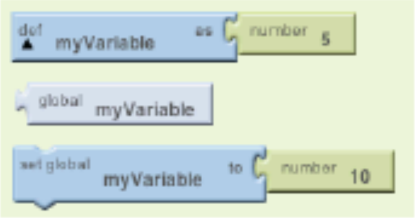
You may give your procedure a name, e.g. "restartGame", and then place the code that should be executed when the procedure is run inside the block.



You may call the procedure you have defined from within your blocks code.

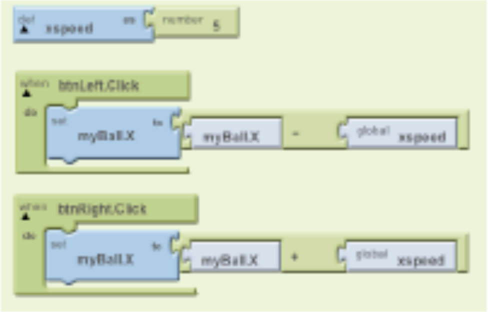


Click anywhere on the screen, and a colorful menu should pop up. On the "Define..." tab, click "variable".



You must define your variable using a "def" block. After that, the value of the variable can be read or modified as many times as is necessary.

Declare a variable that represents the 'speed' of the sprite, i.e. how many pixels to move each time the button is pressed. Then use this variable to update the position of the sprite in the buttons' click handlers.

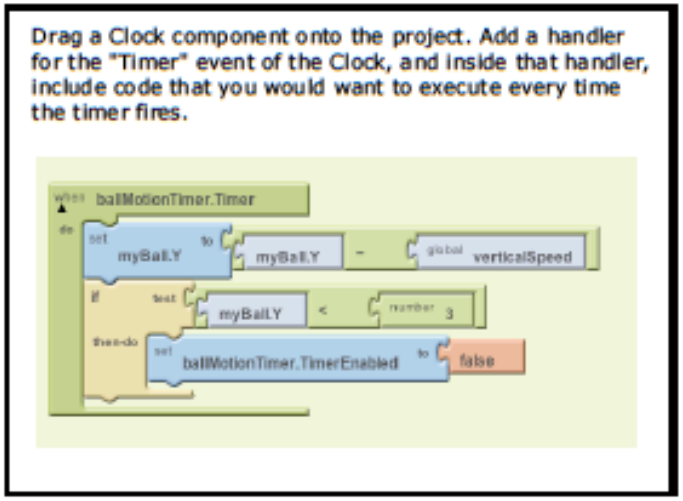


Implementing Jumping and Gravity

Moving Sprites with Clocks

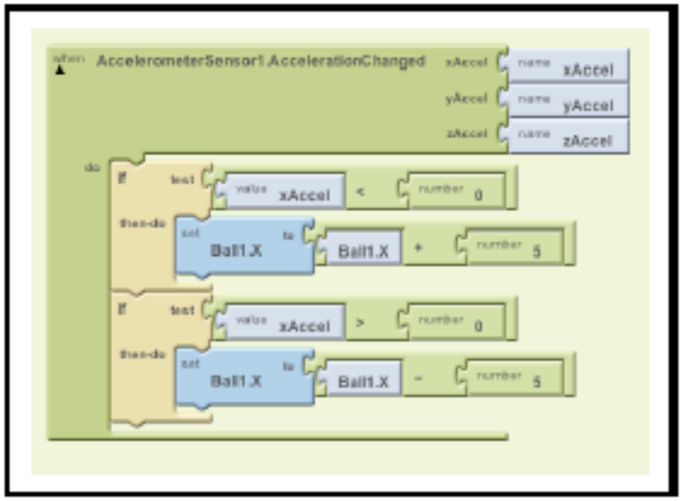
Using AnimatedSprite Component

Moving Sprites with Accelerometer

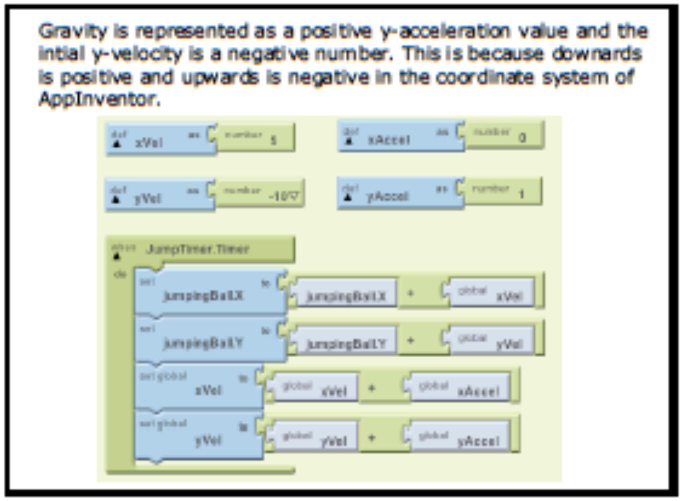


Drag a Clock component onto the project. Add a handler for the "Timer" event of the Clock, and inside that handler, include code that you would want to execute every time the timer fires.

```
when ballMotionTimer.Timer
do  set myBall.Y to  myBall.Y - global verticalSpeed
    if    test  myBall.Y < number 3
    then-do  set ballMotionTimer.TimerEnabled to false
```



```
when AccelerometerSensor1.AccelerationChanged  xAccel  name xAccel
                                               yAccel  name yAccel
                                               zAccel  name zAccel
do  if    test  value xAccel < number 0
    then-do  set Ball1.X to  Ball1.X + number 5
    if    test  value xAccel > number 0
    then-do  set Ball1.X to  Ball1.X - number 5
```

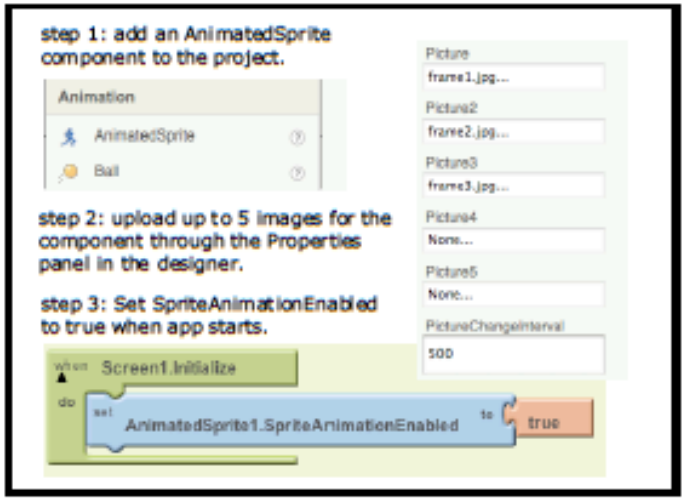


Gravity is represented as a positive y-acceleration value and the intial y-velocity is a negative number. This is because downwards is positive and upwards is negative in the coordinate system of AppInventor.

```
def xVel as number 5          def xAccel as number 0

def yVel as number -10        def yAccel as number 1

when JumpTimer.Timer
do  set jumpingBallX to  jumpingBallX + global xVel
    set jumpingBallY to  jumpingBallY + global yVel
    set global xVel to  global xVel + global xAccel
    set global yVel to  global yVel + global yAccel
```



step 1: add an AnimatedSprite component to the project.

step 2: upload up to 5 images for the component through the Properties panel in the designer.

Picture
frame1.jpg...

Picture2
frame2.jpg...

Picture3
frame3.jpg...

Picture4
None...

Picture5
None...

PictureChangeInterval
500

step 3: Set SpriteAnimationEnabled to true when app starts.

```
when Screen1.Initialize
do  set AnimatedSprite1.SpriteAnimationEnabled to true
```

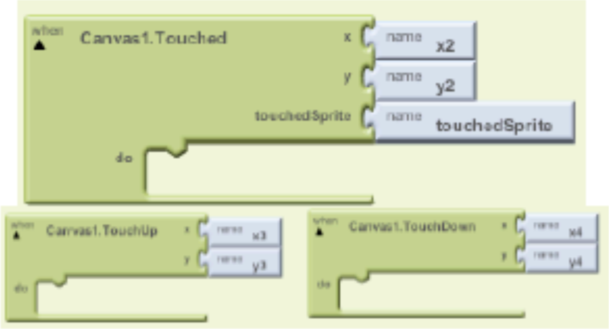
Detecting if Phone is Close to Objects

Touch Events on Canvas and Sprites

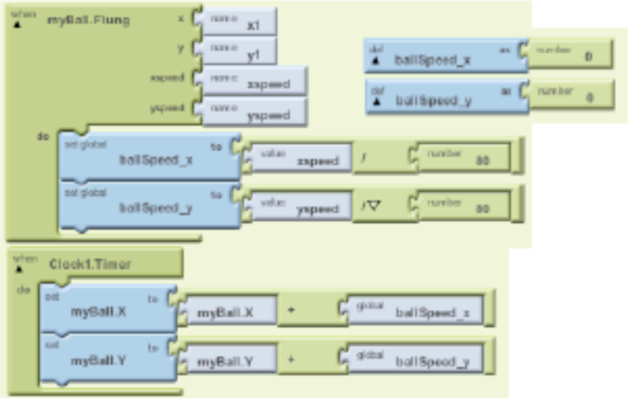
Playing Sounds in your App

Detecting Fling Gestures

There are three types of touch events that can be handled for Canvas and Sprite objects. The "TouchDown" event is triggered as soon as a user's finger touches down on the object, "TouchUp" is triggered when the finger lifts off the object, and "Touched" is triggered when the object is tapped (when TouchDown and TouchUp happen quickly in succession).

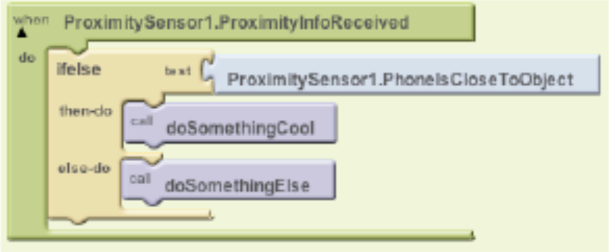


A fling is a quick swipe gesture on the screen. The xspeed and yspeed arguments indicate how fast the flung object should move in the x and y directions.

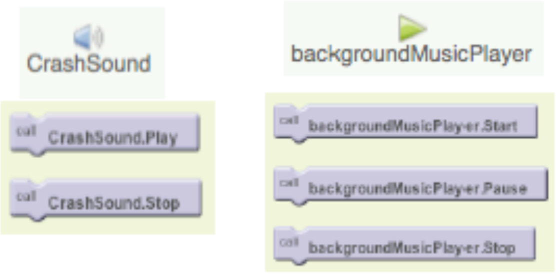

when myBall.Flung
x name x1
y name y1
xspeed name xspeed
yspeed name yspeed
do
set global ballSpeed_x to value xspeed / number 80
set global ballSpeed_y to value yspeed / number 80
def ballSpeed_x as number 0
def ballSpeed_y as number 0
when Clock1.Timer
do
set myBall.X to myBall.X + global ballSpeed_x
set myBall.Y to myBall.Y + global ballSpeed_y

step 1: Drag a ProximitySensor object onto the project.

step 2: In the handler for ProximityInfoReceived event, check the value of the PhoneIsCloseToObject property.


when ProximitySensor1.ProximityInfoReceived
do
ifelse test ProximitySensor1.PhoneIsCloseToObject
then-do call doSomethingCool
else-do call doSomethingElse

There are two components in AppInventor that can play audio. **Sound** components are for playing short audio files, whereas the **Player** component is for playing longer tracks, e.g. background music.


CrashSound
call CrashSound.Play
call CrashSound.Stop
backgroundMusicPlayer
call backgroundMusicPlayer.Start
call backgroundMusicPlayer.Pause
call backgroundMusicPlayer.Stop

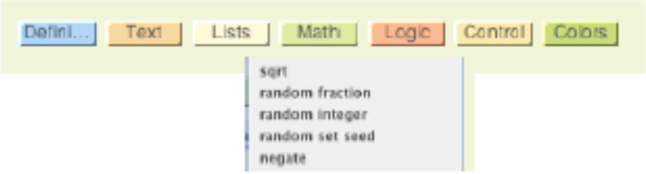
Layouts and Centering Components

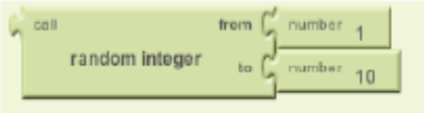
Generating Random Numbers

Collision Detection

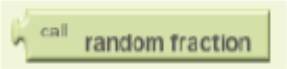
Detecting if Phone is Shaking

Click anywhere on the screen, and a colorful menu should pop up. On the "Math..." tab, click "random fraction" or "random integer".



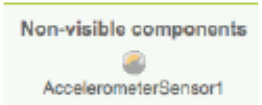


generates a random number between the two specified numbers (inclusive)

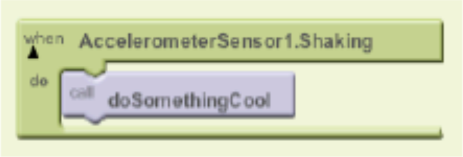


generates a random decimal value between 0 and 1.

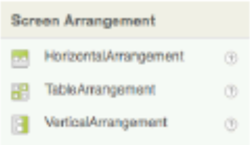
Step 1: drag an Accelerometer onto the project.



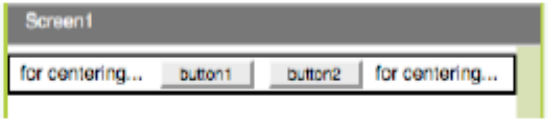
Step 2: in the blocks code, handle the "Shaking" event of the Accelerometer component.



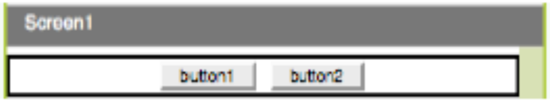
Step 1: Choose from 3 available screen arrangements.



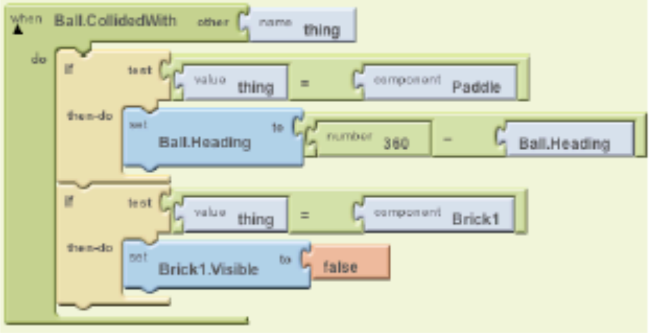
Step 2: Drag multiple components into the arrangement. In order to center them, drag a Label component onto both ends of the arrangement.



Step 3: set the Width of each Label to "Fill parent" and the TextColor to "None". Set the Width of the entire arrangement to "FillParent" as well.



For a game such as brick breaker (where you direct a ball using a paddle, trying to hit bricks), you can check collision by handling the "CollidedWith" event on the Ball component:

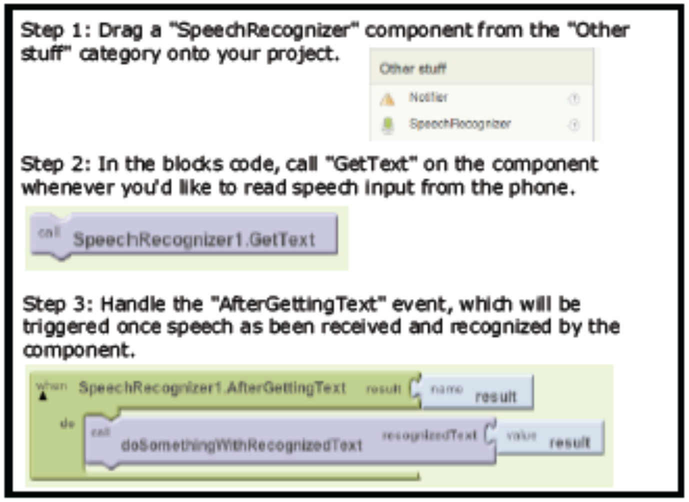

when Ball.CollidedWith other name thing
do
if test value thing = component Paddle
then-do set Ball.Heading to number 360 - Ball.Heading
if test value thing = component Brick1
then-do set Brick1.Visible to false

Creating your own Colors

Speech Recognition

Drawing on the Canvas

Working with Lists of Components


Step 1: Drag a "SpeechRecognizer" component from the "Other stuff" category onto your project.

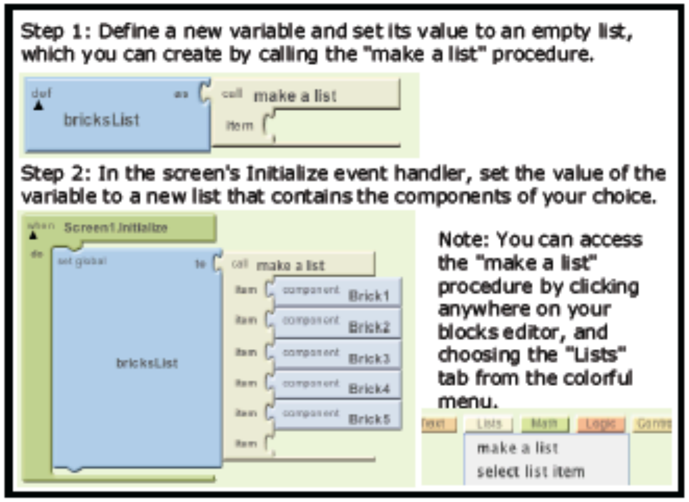
Other stuff
Notifier
SpeechRecognizer

Step 2: In the blocks code, call "GetText" on the component whenever you'd like to read speech input from the phone.

call SpeechRecognizer1.GetText

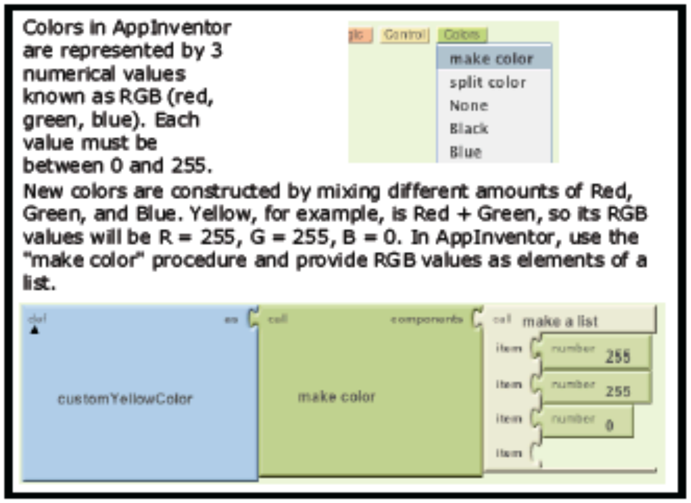
Step 3: Handle the "AfterGettingText" event, which will be triggered once speech as been received and recognized by the component.

when SpeechRecognizer1.AfterGettingText result name result
do call doSomethingWithRecognizedText recognizedText value result


Step 1: Define a new variable and set its value to an empty list, which you can create by calling the "make a list" procedure.

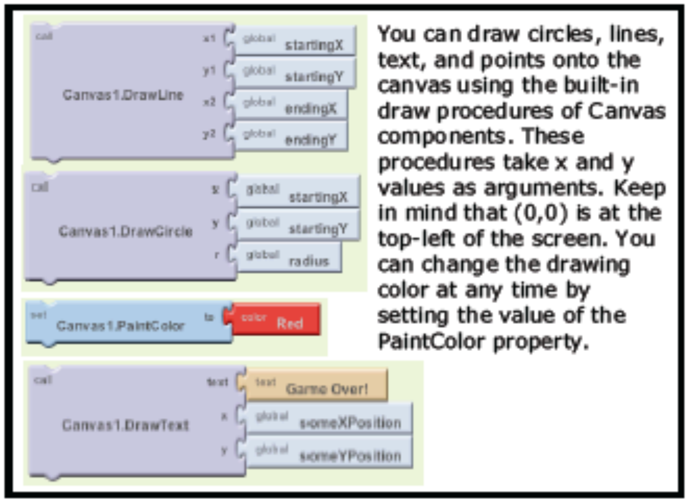
Step 2: In the screen's Initialize event handler, set the value of the variable to a new list that contains the components of your choice.

Note: You can access the "make a list" procedure by clicking anywhere on your blocks editor, and choosing the "Lists" tab from the colorful menu.


Colors in AppInventor are represented by 3 numerical values known as RGB (red, green, blue). Each value must be between 0 and 255.

New colors are constructed by mixing different amounts of Red, Green, and Blue. Yellow, for example, is Red + Green, so its RGB values will be R = 255, G = 255, B = 0. In AppInventor, use the "make color" procedure and provide RGB values as elements of a list.

def customYellowColor as call make color components call make a list
item number 255
item number 255
item number 0
item



You can draw circles, lines, text, and points onto the canvas using the built-in draw procedures of Canvas components. These procedures take x and y values as arguments. Keep in mind that (0,0) is at the top-left of the screen. You can change the drawing color at any time by setting the value of the PaintColor property.