

RAMP

Rowan Academy of Mobile Programming

Bowling Game

Agenda – Day 2

- Creating the Bowling Game
- Lunch Break
- Complete the Bowling Game

Bowling Game Description

In this tutorial, you will write an app that is a bowling game. In this app, you are prompted with a Welcome Screen that allows you to play the game or view the current high score that was achieved. You have ten frames to get the highest score you can by flicking the bowling ball down the lane and hitting the pins. Along the way, concepts of Computer Science and Android will be taught.

Android & Computer Science Concepts Covered in the Bowling Game

This Android App will include the following Computer Science concepts and Android principles:

- An algorithm - a precise sequence of instructions for a process that is executed
- Developing abstractions (logic and control statements)
- Information processing to gain insight
- Variables and Data Storage

Screen 1

The screenshot displays the MIT App Inventor 2 Beta web interface. At the top, the title bar shows "MIT App Inventor 2 Beta" and navigation menus for "Projects", "Connect", "Build", "Help", "My Projects", "Gallery", "Guide", "Report an Issue", "English", and a user email "john.robinson616@gmail.com". Below the title bar, the project name "Bowling2016" is shown, along with "Screen1" and buttons for "Add Screen ..." and "Remove Screen". On the right side of this bar are "Designer" and "Blocks" tabs.

The main workspace is divided into four panels:

- Palette:** Contains categories for "User Interface" (Button, CheckBox, DatePicker, Image, Label, ListPicker, ListView, Notifier, PasswordTextBox, Slider, Spinner, TextBox, TimePicker, WebViewer), "Layout", "Media", and "Drawing and Animation".
- Viewer:** Shows a mobile device preview of the "Bowling Game" screen. The screen has a blue background with the word "Bowling" in large white text. Below it, there are three lines of text: "Rules:", "1. No Spares! Just Strikes or the amount of pins hit.", "2. Strikes count next frame as double, but just once unlike normal bowling score keeping.", and "3. The tenth frame is just like all the over frames." At the bottom of the screen are three buttons: "Start Game", "High Score", and "Quit". The device status bar at the top shows signal strength, Wi-Fi, battery, and the time "9:48".
- Components:** A tree view showing the hierarchy of components on the screen: "Screen1" contains "HorizontalArrangement1", which contains "HorizontalArrangement2", which contains "StartGame", "HorizontalArrangemen", and "HighScore". Below this is a "Quit" component.
- Properties:** Shows the properties for the selected "Screen1" component. Properties include: "AboutScreen" (text input), "AlignHorizontal" (set to "Center"), "AlignVertical" (set to "Top"), "AppName" (set to "BowlingGame"), "BackgroundColor" (set to "White"), "BackgroundImage" (set to "WelcomeScreenFix.png..."), "CloseScreenAnimation" (set to "Default"), "Icon" (set to "None..."), "OpenScreenAnimation" (set to "Default"), "ScreenOrientation" (set to "Unspecified"), and "Scrollable".

Screen 1 Component Settings

Properties

HorizontalArrangement1

AlignHorizontal
Left ▾

AlignVertical
Top ▾

BackgroundColor
 None

Height
310 pixels...

Width
Fill parent...

Image
None...

Visible

Properties

HorizontalArrangement2

AlignHorizontal
Center ▾

AlignVertical
Center ▾

BackgroundColor
 None

Height
50 pixels...

Width
Fill parent...

Image
None...

Visible

Properties

HorizontalArrangement3

AlignHorizontal
Left ▾

AlignVertical
Top ▾

BackgroundColor
 None

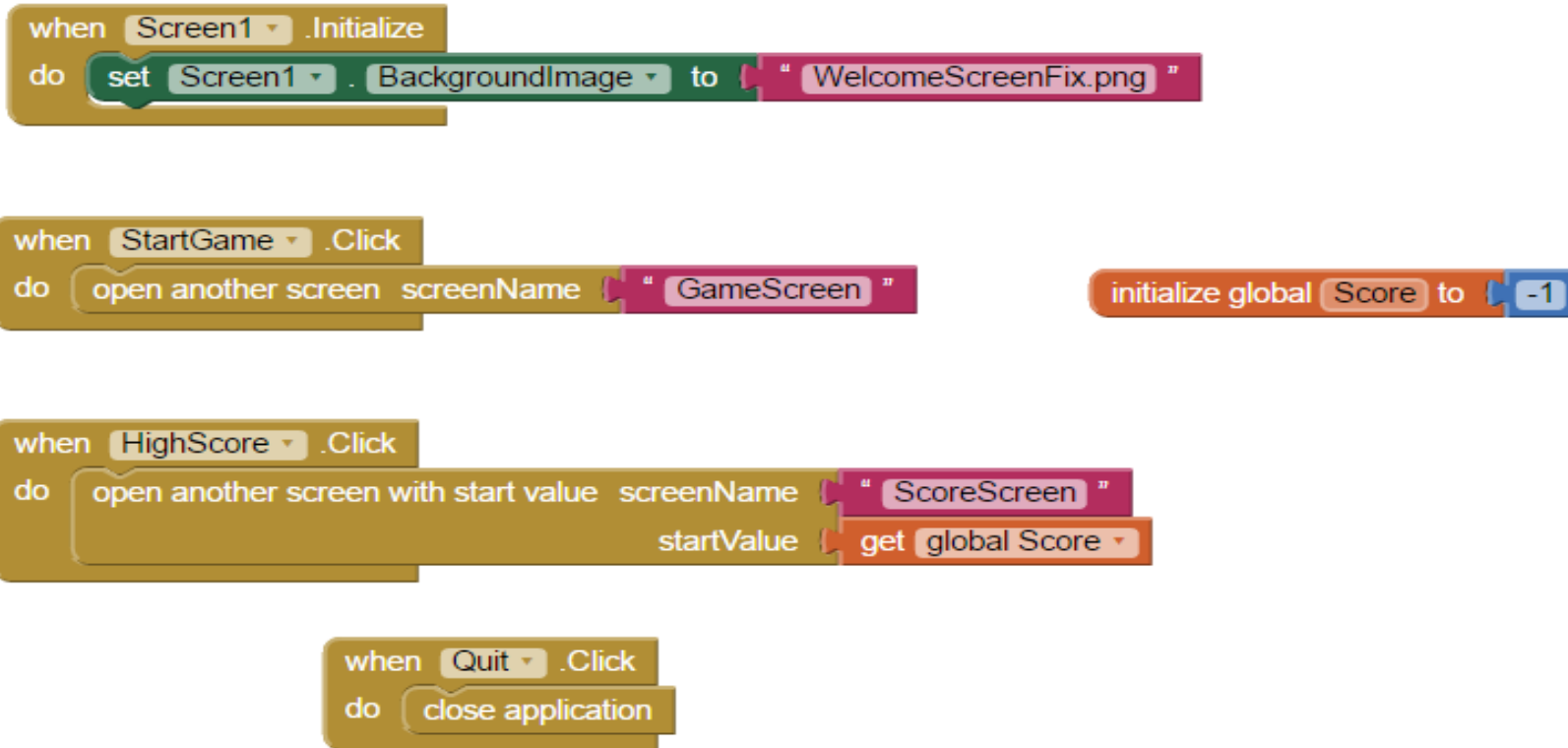
Height
Automatic...

Width
10 pixels...

Image
None...

Visible

Screen1 Code



The code consists of four event-driven blocks:

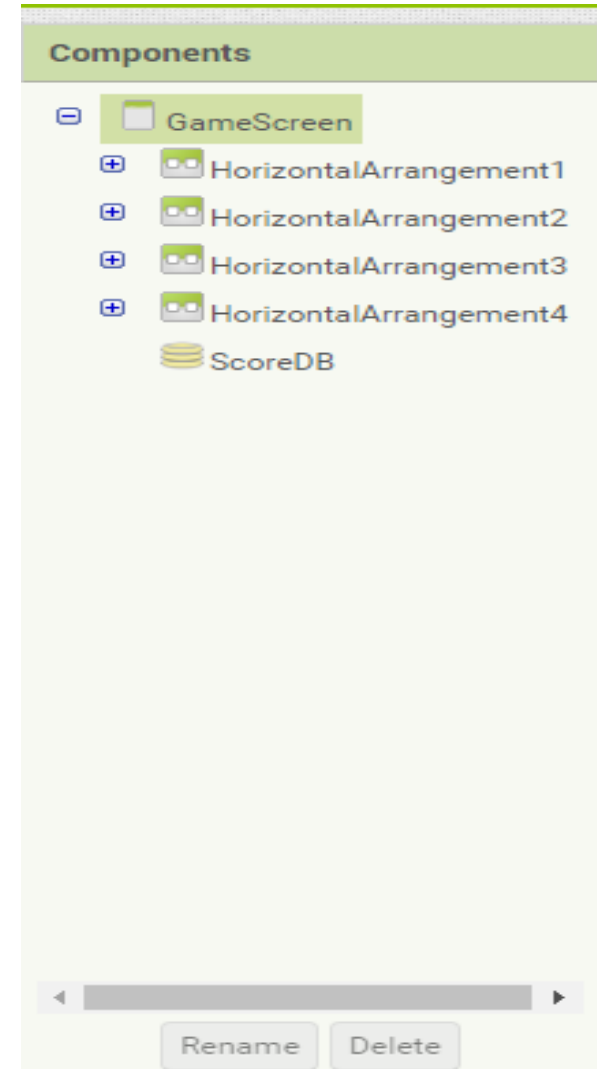
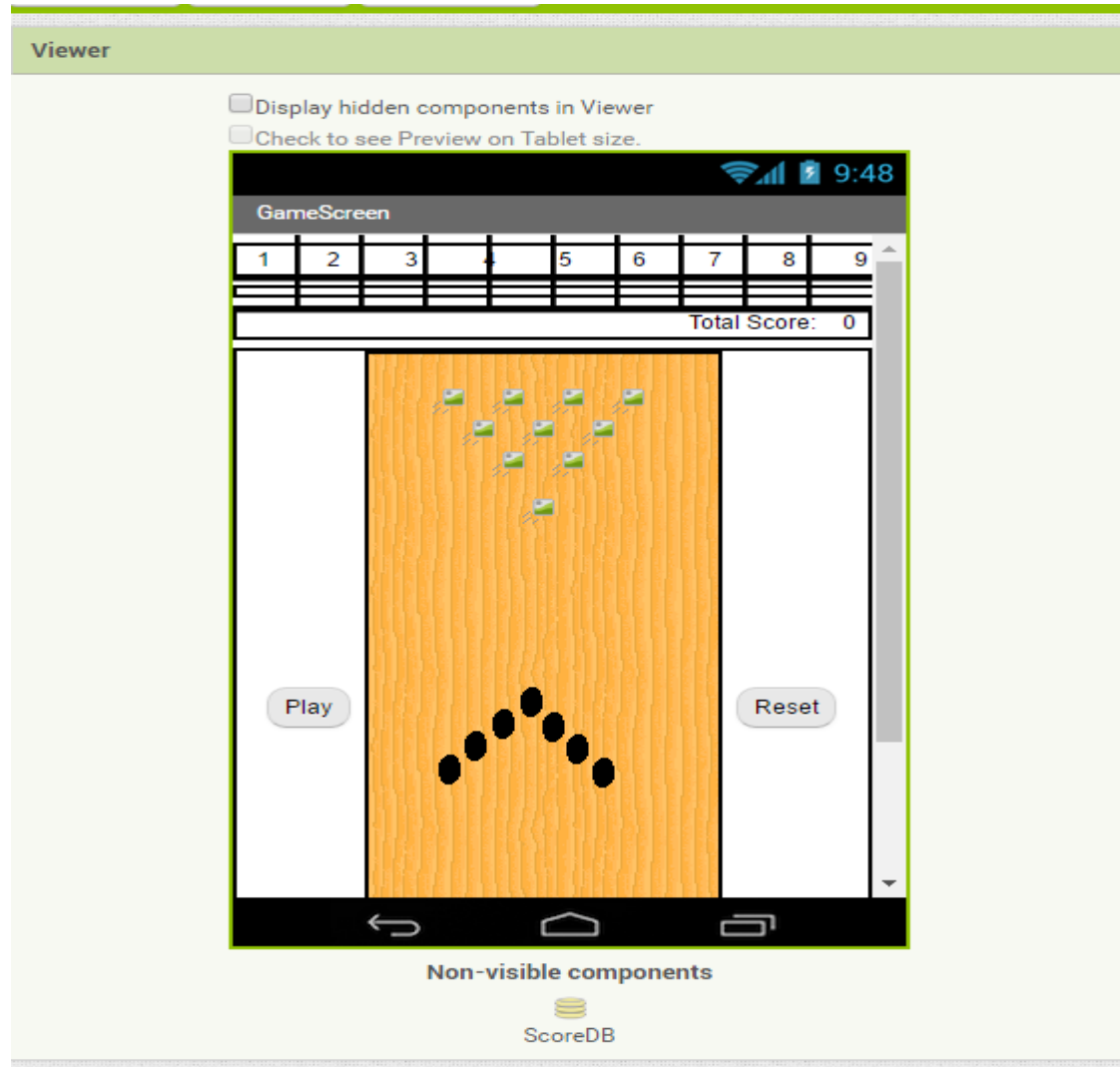
- when Screen1 Initialize**
 - do **set Screen1 BackgroundImage to "WelcomeScreenFix.png"**
- when StartGame Click**
 - do **open another screen screenName "GameScreen"**
 - initialize global Score to -1**
- when HighScore Click**
 - do **open another screen with start value screenName "ScoreScreen"**
 - startValue get global Score**
- when Quit Click**
 - do **close application**



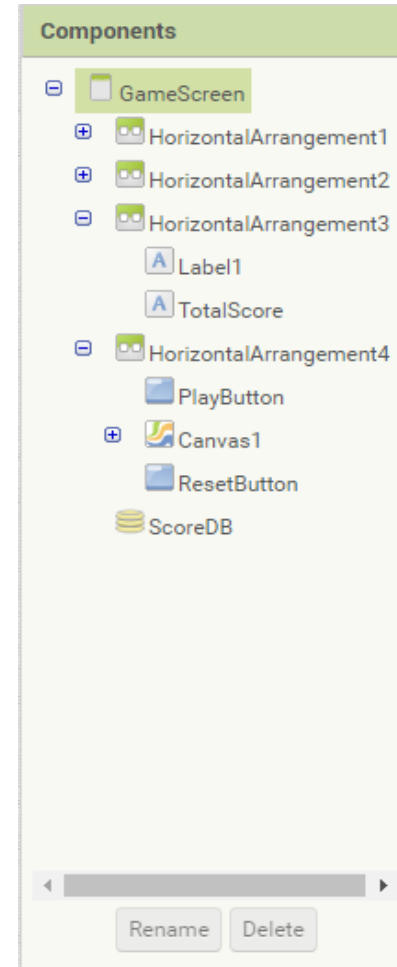
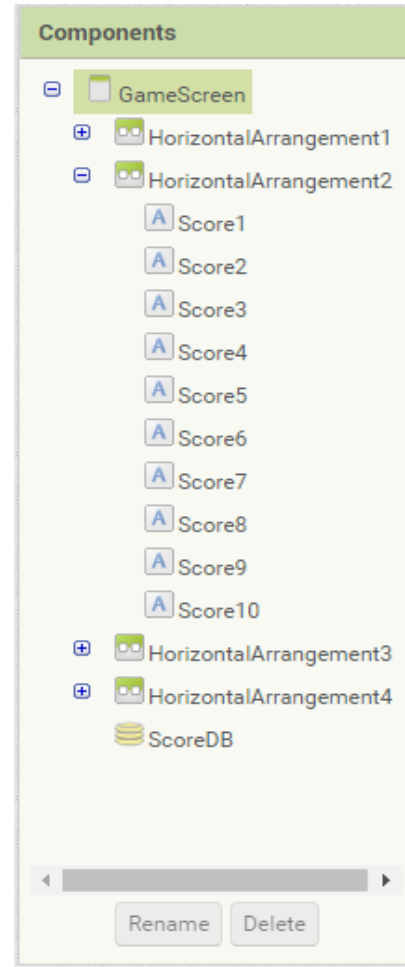
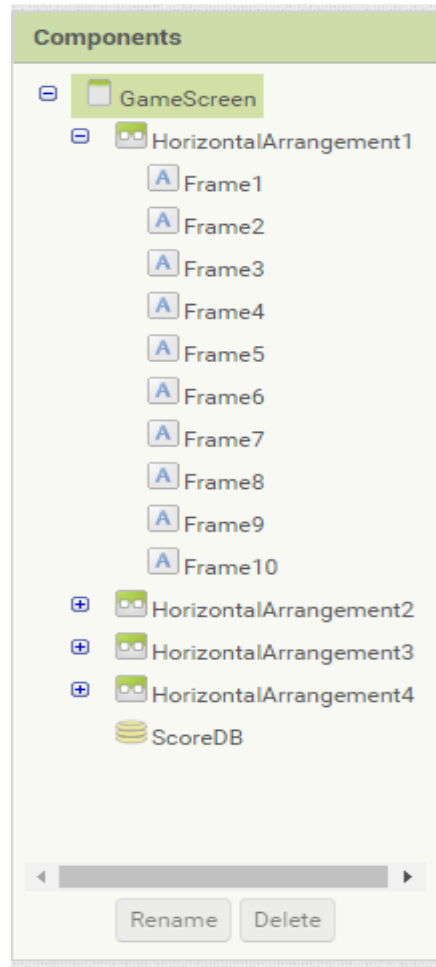
0 0

Show Warnings

Game Screen



Game Screen Components



Game Screen Components Settings

Properties

HorizontalArrangement1

AlignHorizontal
Left ▾

AlignVertical
Top ▾

BackgroundColor
 None

Height
Automatic...

Width
320 pixels...

Image
None...

Visible

Properties

HorizontalArrangement2

AlignHorizontal
Left ▾

AlignVertical
Top ▾

BackgroundColor
 None

Height
Automatic...

Width
320 pixels...

Image
None...

Visible

Properties

HorizontalArrangement4

AlignHorizontal
Right ▾

AlignVertical
Top ▾

BackgroundColor
 None

Height
Automatic...

Width
Fill parent...

Image
None...

Visible

Properties

HorizontalArrangement3

AlignHorizontal
Center ▾

AlignVertical
Center ▾

BackgroundColor
 None

Height
Fill parent...

Width
Fill parent...

Image
None...

Visible

Game Screen Component Settings

Properties

Frame1

BackgroundColor
 None

FontBold

FontItalic

FontSize
14.0

FontTypeface
sans serif ▾

HasMargins

Height
Automatic...

Width
22 pixels...

Text
1

TextAlignment
center ▾

TextColor
 Black

Visible

	Text	Width
Frame2	2	30 pixels
Frame3	3	30 pixels
Frame4	4	30 pixels
Frame5	5	28 pixels
Frame6	6	28 pixels
Frame7	7	30 pixels
Frame8	8	27 pixels
Frame9	9	27 pixels
Frame10	10	30 pixels

Game Screen Component Settings

Properties

Score1

BackgroundColor
 None

FontBold

FontItalic

FontSize
14.0

FontTypeface
sans serif ▾

HasMargins

Height
Automatic...

Width
30 pixels...

Text

TextAlignment
left ▾

TextColor
■ Black

Visible

	Width
Score2	30 pixels
Score3	27 pixels
Score4	30 pixels
Score5	27 pixels
Score6	27 pixels
Score7	27 pixels
Score8	30 pixels
Score9	30 pixels
Score10	30 pixels

Game Screen Component Settings

Properties

Label1

BackgroundColor
 None

FontBold

FontItalic

FontSize
14.0

FontTypeface
sans serif ▾

HasMargins

Height
Automatic...

Width
Automatic...

Text
Total Score:

TextAlignment
left ▾

TextColor
■ Black

Visible

Properties

TotalScore

BackgroundColor
 None

FontBold

FontItalic

FontSize
14.0

FontTypeface
sans serif ▾

HasMargins

Height
Automatic...

Width
Automatic...

Text
0

TextAlignment
left ▾

TextColor
■ Black

Visible

Properties

PlayButton

BackgroundColor
■ Default

Enabled

FontBold

FontItalic

FontSize
14.0

FontTypeface
default ▾

Height
Automatic...

Width
Automatic...

Image
None...

Shape
default ▾

ShowFeedback

Text
Play

TextAlignment
center ▾

TextColor
■ Default

Visible

Game Screen Component Settings

Properties

ResetButton

BackgroundColor
 Default

Enabled

FontBold

FontItalic

FontSize
14.0

FontTypeface
default ▾

Height
Automatic...

Width
Automatic...

Image
None...

Shape
default ▾

ShowFeedback

Text
Reset

TextAlignment
center ▾

TextColor
 Default

Visible

Properties

Canvas1

BackgroundColor
 White

BackgroundImage
lane.png...

FontSize
14.0

Height
450 pixels...

Width
175 pixels...

LineWidth
2.0

PaintColor
 None

TextAlignment
left ▾

Visible

Properties

BowlingBall

Enabled

Heading
0

Interval
100

PaintColor
 Blue

Radius
10

Speed
0.0

Visible

X
75

Y
415

Z
1.0

Game Screen Component Settings

Properties

Pin1

Enabled

Heading

Height

Width

Interval

Picture

Rotates

Speed

Visible

X

Y

Z

	X	Y
Pin2	60	60
Pin3	90	60
Pin4	45	40
Pin5	75	40
Pin6	105	40
Pin7	30	20
Pin8	60	20
Pin9	90	20
Pin10	120	20

Game Screen Code



Game Screen Code

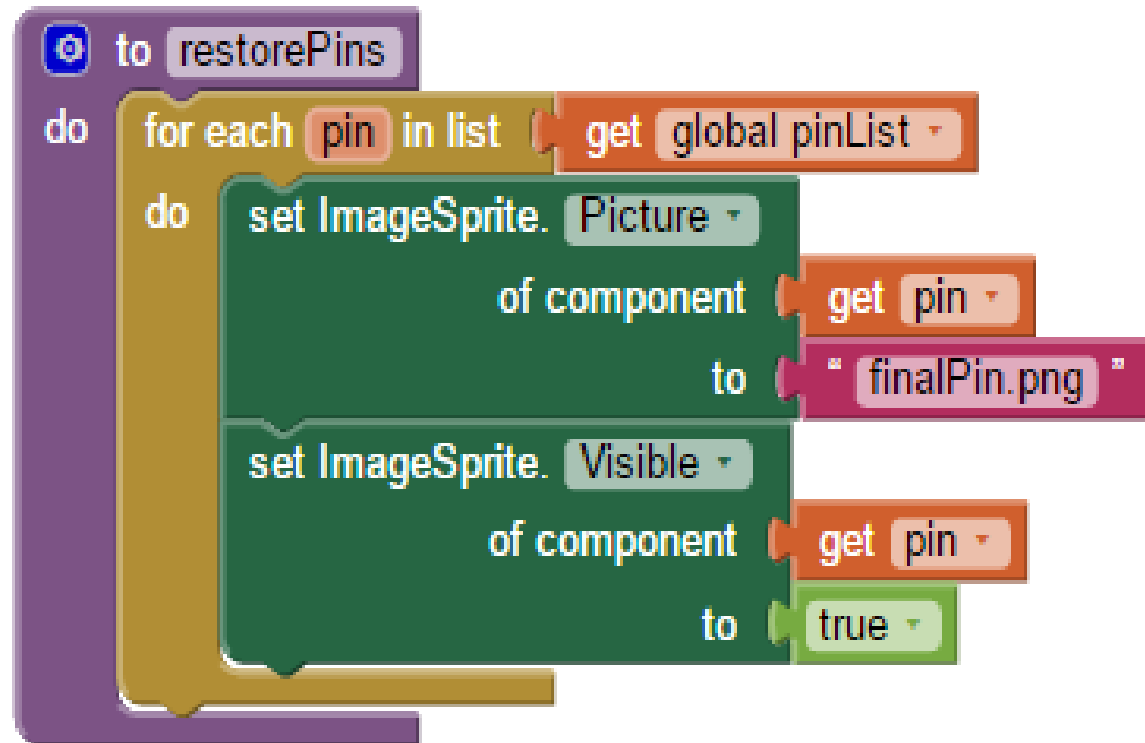
```
when GameScreen.Initialize
do
  add items to list list get global ScoreList
  item Score1
  item Score2
  item Score3
  item Score4
  item Score5
  item Score6
  item Score7
  item Score8
  item Score9
  item Score10
  item

  set GameScreen.BackgroundImage to "gameBackground.png"

  add items to list list get global pinList
  item Pin10
  item Pin9
  item Pin8
  item Pin7
  item Pin6
  item Pin5
  item Pin4
  item Pin3
  item Pin2
  item Pin1
  item

  set BowlingBall.Visible to false
  call restorePins
  to restorePins
  do
```

Game Screen Code



```
to restorePins
do
  for each pin in list
    get global pinList
  do
    set ImageSprite. Picture
      of component
      to
      "finalPin.png"
    set ImageSprite. Visible
      of component
      to
      true
```

The image shows a Scratch code block for a function named 'restorePins'. The function starts with a 'do' block containing a 'for each pin in list' loop. Inside the loop, there are two 'do' blocks. The first 'do' block contains a 'set ImageSprite. Picture of component to' block with a 'get global pinList' block and a 'get pin' block. The second 'do' block contains a 'set ImageSprite. Visible of component to' block with a 'get pin' block and a 'true' block. The 'set ImageSprite. Picture of component to' block has a dropdown menu set to 'Picture' and the value 'finalPin.png'. The 'set ImageSprite. Visible of component to' block has a dropdown menu set to 'Visible' and the value 'true'.

Game Screen Code

```
when Canvas1 .Touched
  x y touchedAnySprite
do
  if [get y] > 380
  then
```

```
when BowlingBall .Flung
  x y speed heading xvel yvel
do
  set BowlingBall .Heading to [get heading]
  set BowlingBall .Speed to [get speed] x 6
```

Game Screen Code

```
when BowlingBall .CollidedWith other do if call checkForStrike and get global shotCounter = 0 then set global frameScore to 10 call displayScore frame select list item list get global ScoreList index get global FrameCount set global strike to false set global shotCounter to 1 if not get global strike then call pinsfalls pinHit Pin1 for each hitPin in list get global pinList do call pinsfalls pinHit get hitPin
```

```
to checkForStrike result
```

```
to displayScore frame do
```

```
to pinsfalls pinHit do
```

Game Screen Code

```
to checkForStrike
  result ← do
    if BowlingBall .CollidingWith other Pin1
      then
        if BowlingBall .Speed ≥ 13
          then
            set global strike to true
            for each strikePin in list get global pinList
              do
                set ImageSprite.Visible of component get strikePin to false
          result ← get global strike
```

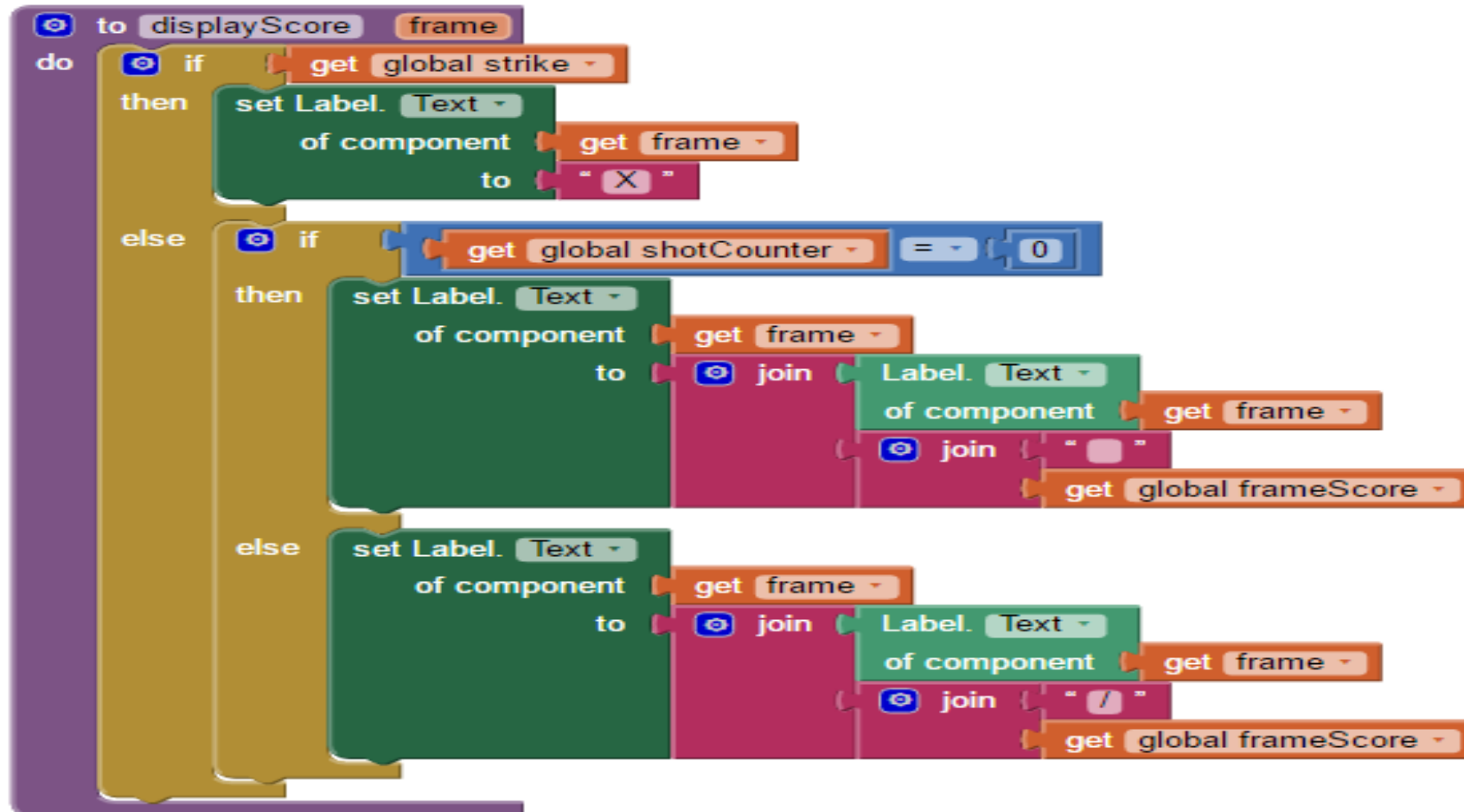
The code is a Scratch script for a function named 'checkForStrike'. It starts with a 'do' block that contains three nested 'if' blocks. The first 'if' block checks if a 'BowlingBall' is colliding with 'Pin1'. If true, it enters a 'then' block with a second 'if' block that checks if the 'BowlingBall' speed is greater than or equal to 13. If true, it enters another 'then' block with a 'set global strike to true' block, followed by a 'for each' loop over 'strikePin' in 'global pinList'. Inside the loop, it sets the 'Visible' property of the 'ImageSprite' of the 'strikePin' component to 'false'. Finally, the 'do' block returns the value of 'global strike'.

Game Screen Code

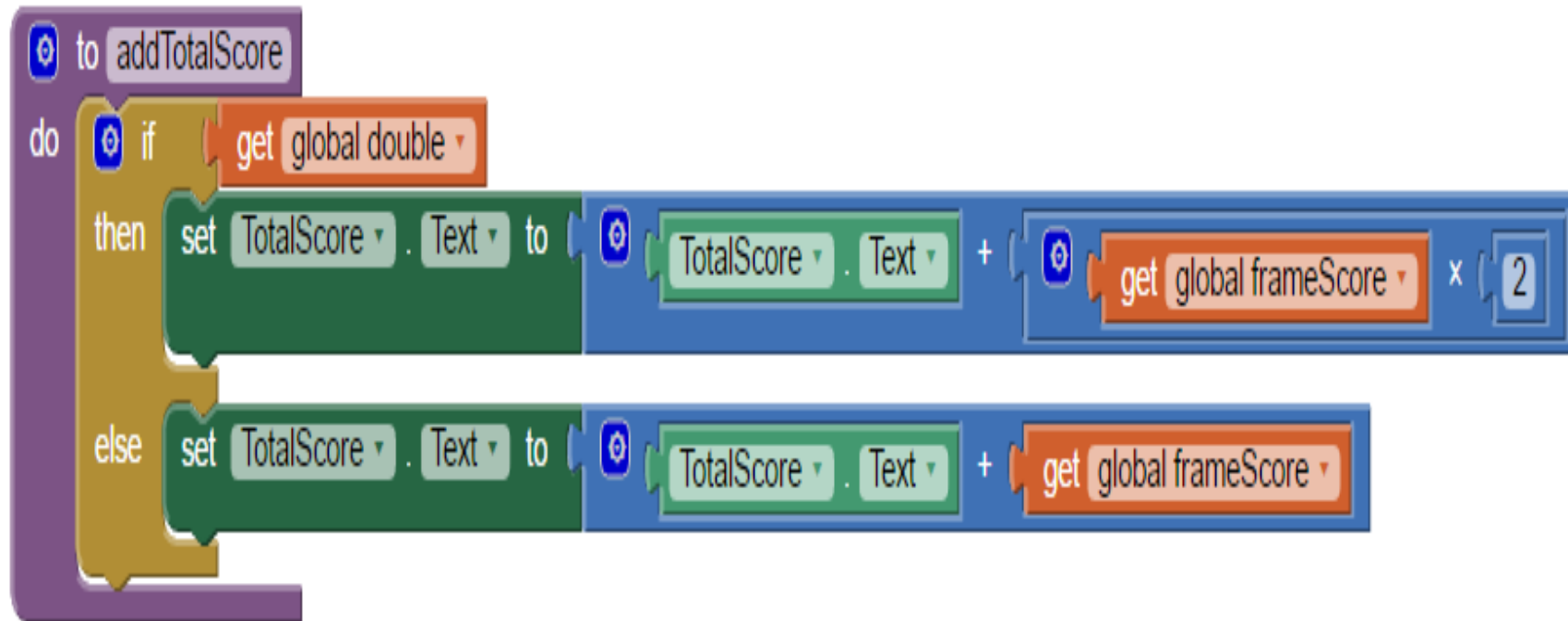
```
to pinsfalls pinHit
do
  if call BowlingBall .CollidingWith other get pinHit
  then
    set ImageSprite. Visible
    of component get pinHit
    to false
    set global frameScore to get global frameScore + 1
```

The image shows a Scratch script for a bowling game. It starts with a 'to pinsfalls pinHit' block. Inside a 'do' loop, there is an 'if' block that checks if a 'BowlingBall' is colliding with another object (retrieved via 'get pinHit'). If the condition is true, the 'then' block contains four actions: setting the 'Visible' property of the 'ImageSprite' of the 'get pinHit' component to 'false', and then setting the 'global frameScore' to the current 'global frameScore' plus 1.

Game Screen Code



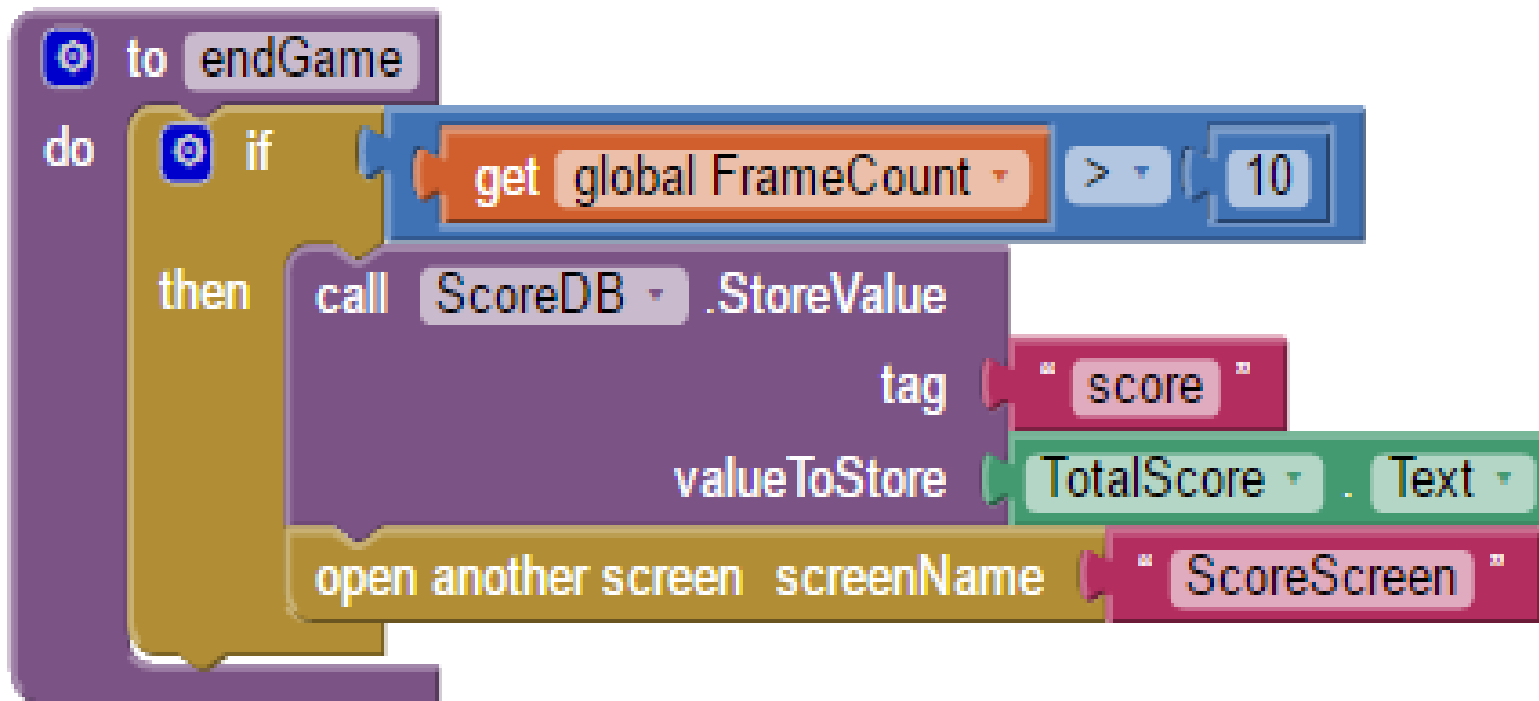
Game Screen Code



Game Screen Code

```
to resetBall
do
  call BowlingBall .MoveTo
    x get global ballDefaultX
    y get global ballDefaultY
  set BowlingBall . Heading to 0
  set BowlingBall . Speed to 0
  if get global shotCounter = 2
  then
    set global shotCounter to 0
    set global FrameCount to get global FrameCount + 1
    call restorePins
```

Game Screen Code



```
to endGame
do
  if (get global FrameCount > 10)
  then
    call ScoreDB .StoreValue
      tag "score"
      valueToStore TotalScore . Text
    open another screen screenName "ScoreScreen"
```

The image shows a Scratch code block for a function named 'endGame'. It starts with a 'do' block containing an 'if' block. The 'if' block checks if the 'global FrameCount' is greater than 10. If true, it executes three actions: 'call ScoreDB .StoreValue' with tag 'score' and value 'TotalScore . Text', and 'open another screen screenName' with screen name 'ScoreScreen'.

Game Screen Code

```
when BowlingBall .EdgeReached
  edge
do
  if (get edge == 1 or get edge == 2 or get edge == -4)
  then
    call addTotalScore
    if (get global frameScore != 10)
    then
      call displayScore
      frame select list item list (get global ScoreList) (get global FrameCount)
    else
      set global strike to true
      set global double to true
      set global shotCounter to (get global shotCounter + 1)
      if (get global shotCounter == 2 or not get global strike)
      then
        set global double to false
        set global frameScore to 0
        set global strike to false
        call resetBall
        call endGame
    if (get edge == -1 or get edge == 4 or get edge == -2)
    then
      call resetBall
```

Game Screen Code

```
when PlayButton .Click
do
  set global FrameCount to (get global FrameCount + 1)
  set BowlingBall .Visible to true
  set PlayButton .Enabled to false
```

```
when ResetButton .Click
do
  call restorePins
  call resetBall
  for each var in list (get global ScoreList)
  do
    set Label .Text of component (get var) to ""
  set global shotCounter to 0
  set TotalScore .Text to 0
  set BowlingBall .Visible to false
  set PlayButton .Enabled to true
  set global FrameCount to 0
```

Score Screen

The screenshot displays the Android Studio IDE with a 'ScoreScreen' project. The central 'Viewer' pane shows a mobile app interface with a green background and wood-grain borders. The text 'High Score' and 'Your Score' are displayed, both with a value of '0'. Below the scores are 'Return Home' and 'Quit' buttons. The top status bar shows the time as 9:48. The 'Palette' on the left lists various UI components like 'User Interface', 'Layout', 'Media', etc. The 'Components' pane on the right shows a hierarchical tree of the screen's layout, including 'VerticalArrangement1', 'HorizontalArrangement' containers, and text labels like 'HighScore', 'GameScore', 'ReturnHome', and 'Quit'. The 'Properties' pane on the far right shows settings for the 'ScoreScreen' component, such as 'AboutScreen', 'AlignHorizontal' (Left), 'AlignVertical' (Top), 'BackgroundColor' (White), and 'BackgroundImage' (highscore.jpg...). The 'Media' pane at the bottom right lists image assets like 'WelcomeScreenFix.png', 'finalPin.png', 'gameBackground.png', and 'highscore.jpg'. Below the viewer, 'Non-visible components' lists 'HighScoreDB' and 'ScoreDB'.

Score Screen Component Settings

Properties

VerticalArrangement1

AlignHorizontal
Center ▾

AlignVertical
Top ▾

BackgroundColor
 None

Height
Fill parent...

Width
Fill parent...

Image
None...

Visible

Properties

HorizontalArrangement1

AlignHorizontal
Left ▾

AlignVertical
Top ▾

BackgroundColor
 None

Height
125 pixels...

Width
Automatic...

Image
None...

Visible

Properties

HorizontalArrangement2

AlignHorizontal
Center ▾

AlignVertical
Top ▾

BackgroundColor
 None

Height
Automatic...

Width
Fill parent...

Image
None...

Visible

Score Screen Component Settings

Properties

HorizontalArrangement3

AlignHorizontal
Left ▾

AlignVertical
Top ▾

BackgroundColor
 None

Height
190 pixels...

Width
Automatic...

Image
None...

Visible

Properties

HorizontalArrangement4

AlignHorizontal
Center ▾

AlignVertical
Top ▾

BackgroundColor
 None

Height
Automatic...

Width
Fill parent...

Image
None...

Visible

Properties

HorizontalArrangement5

AlignHorizontal
Left ▾

AlignVertical
Top ▾

BackgroundColor
 None

Height
50 pixels...

Width
Automatic...

Image
None...

Visible

Score Screen Component Settings

Properties

HorizontalArrangement6

AlignHorizontal
Center ▾

AlignVertical
Top ▾

BackgroundColor
 None

Height
Automatic...

Width
Fill parent...

Image
None...

Visible

Properties

HighScore

BackgroundColor
 None

FontBold

FontItalic

FontSize
25.0

FontTypeface
sans serif ▾

HasMargins

Height
Automatic...

Width
Automatic...

Text
0

TextAlignment
left ▾

TextColor
 Black

Visible

Properties

GameScore

BackgroundColor
 None

FontBold

FontItalic

FontSize
25.0

FontTypeface
default ▾

HasMargins

Height
Automatic...

Width
Automatic...

Text
0

TextAlignment
left ▾

TextColor
 Black

Visible

Properties

returnHome

BackgroundColor
 Default

Enabled

FontBold

FontItalic

FontSize
14.0

FontTypeface
sans serif ▾

Height
Automatic...

Width
Automatic...

Image
None...

Shape
rounded ▾

ShowFeedback

Text
Home

TextAlignment
center ▾

TextColor
 Default

Visible

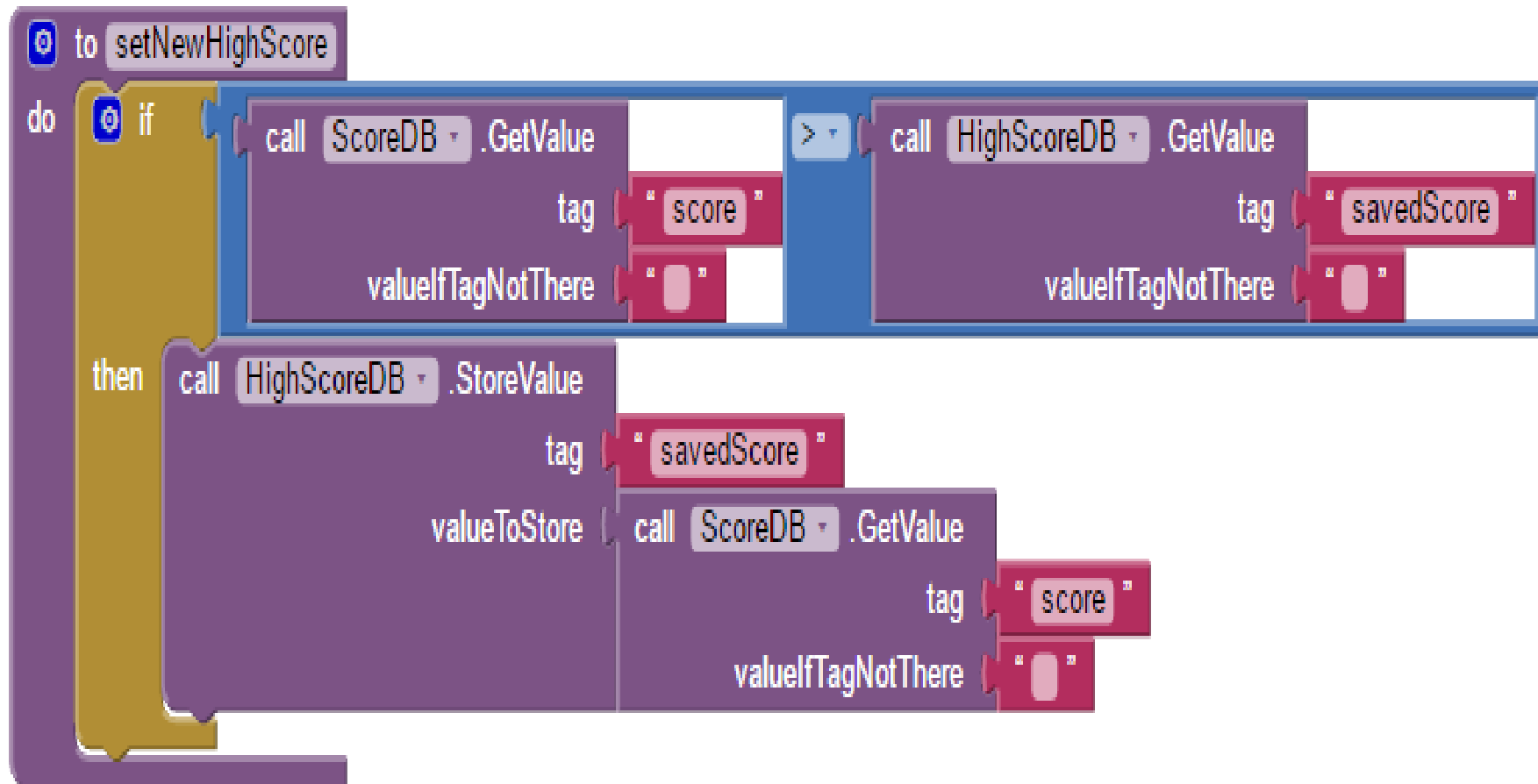
Score Screen Code

```
when ScoreScreen.Initialize
do
  set ScoreScreen.BackgroundImage to "highScore.jpg"
  if
    call HighScoreDB.GetValue
      tag "savedScore"
      valueIfTagNotThere ""
    = ""
  then
    call HighScoreDB.StoreValue
      tag "savedScore"
      valueToStore 0
    call setNewHighScore
    set HighScore.Text to call HighScoreDB.GetValue
      tag "savedScore"
      valueIfTagNotThere ""
    call displayGameScore
  else
    call setNewHighScore
    set HighScore.Text to call HighScoreDB.GetValue
      tag "savedScore"
      valueIfTagNotThere ""
    call displayGameScore
end if
```

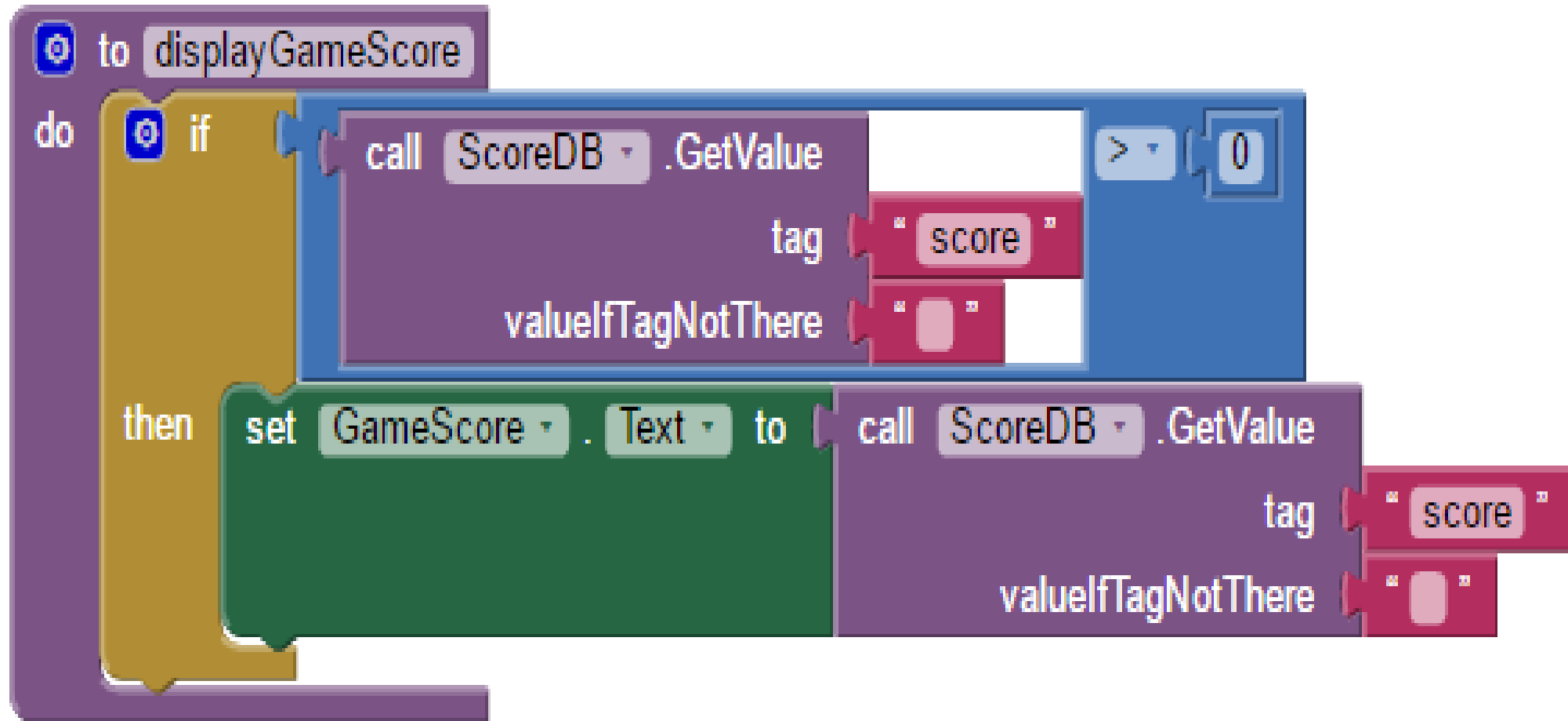
```
to setNewHighScore
do
end
```

```
to displayGameScore
do
end
```

Score Screen Code



Score Screen Code



```
to displayGameScore
do
  if
    call ScoreDB .GetValue
      tag "score"
      valueIfTagNotThere ""
    > 0
  then
    set GameScore . Text to
      call ScoreDB .GetValue
        tag "score"
        valueIfTagNotThere ""
```

The image shows a Scratch script for a function named 'displayGameScore'. The script is contained within a purple 'to' block. Inside, there is a 'do' block containing an 'if' block. The 'if' block has a condition: 'call ScoreDB .GetValue' with a 'tag' of 'score' and a 'valueIfTagNotThere' of an empty string, followed by a '>' operator and a '0' value. If this condition is true, the 'then' block executes, which is a 'set GameScore . Text to' block. This block also contains a 'call ScoreDB .GetValue' block with the same 'tag' and 'valueIfTagNotThere' parameters.

Score Screen Code

