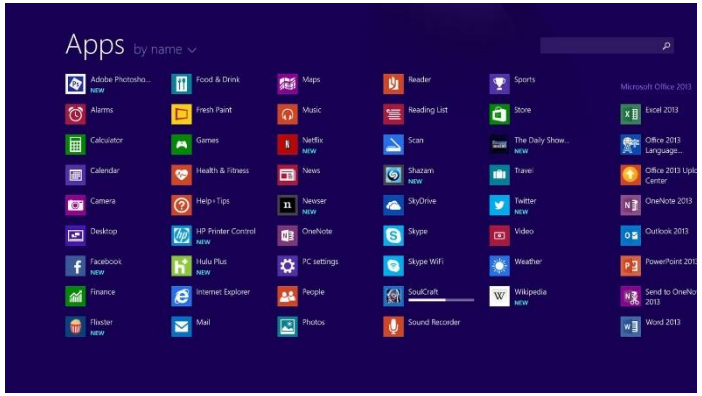
Rowan Academy of Mobile Programming

RAMP



Except as otherwise noted, the content of this document is licensed under the Creative Commons Attribution 3.0 License
http://creativecommons.org/licenses/by/3.0

Agenda – Day 1

- Introduction
- What is a computer program?
- Introducing App Inventor
- Getting hands on with App Inventor
- Lunch Break
- Creating the Mole Mash Game

(cc) BY-NC

Rowan Academy of Mobile Programming (RAMP)

- Mobile application programming can provide an authentic and engaging hook into computer science. The MIT App Inventor is a visual programming environment that enables students with no programming background to build apps for Android mobile devices. We will use this at Rowan University to teach CS Principles to K-12 students and educators by empowering them to create their own mobile apps and engage them personally, as well as infusing energy and excitement into computer science education.

For more information contact:

John Robinson, robinsonj@rowan.edu, 856-256-4778

The College Board and NSF CS Big Ideas

- Big Idea I: Creativity
- Big Idea II: Abstraction
- Big Idea III: Data
- Big Idea IV: Algorithms
- Big Idea V: Programming
- Big Idea VI: Internet
- Big Idea VII: Global Impact

(cc) BY-NC

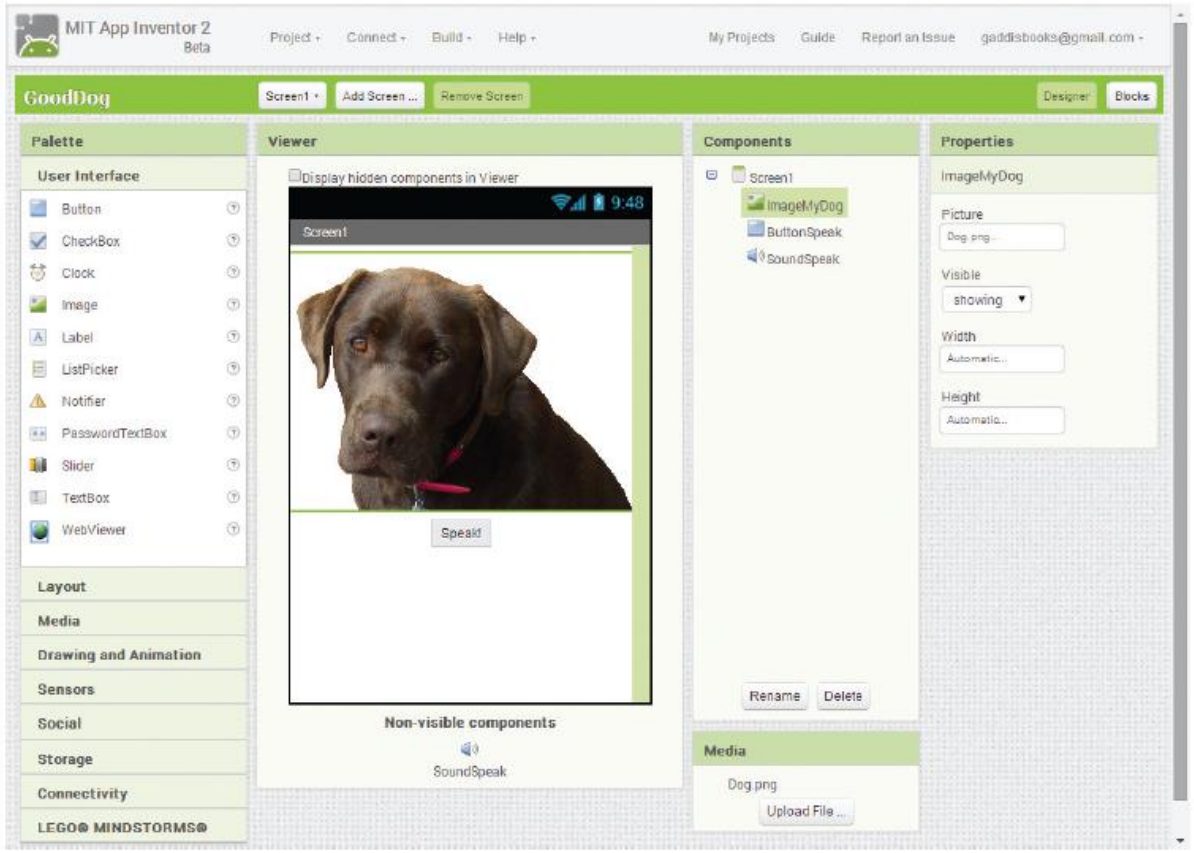
Introduction

- You may quickly and easily create applications or “apps” for android smartphones and tablets.
- With App Inventor, you use a screen designer to visually create an app’s screen (Figure 1-1).
- Then you use a special editor known as the Blocks Editor to create the actions.
- You visually assemble *code blocks* (Figure 1-2).

(cc) BY-NC

Introduction

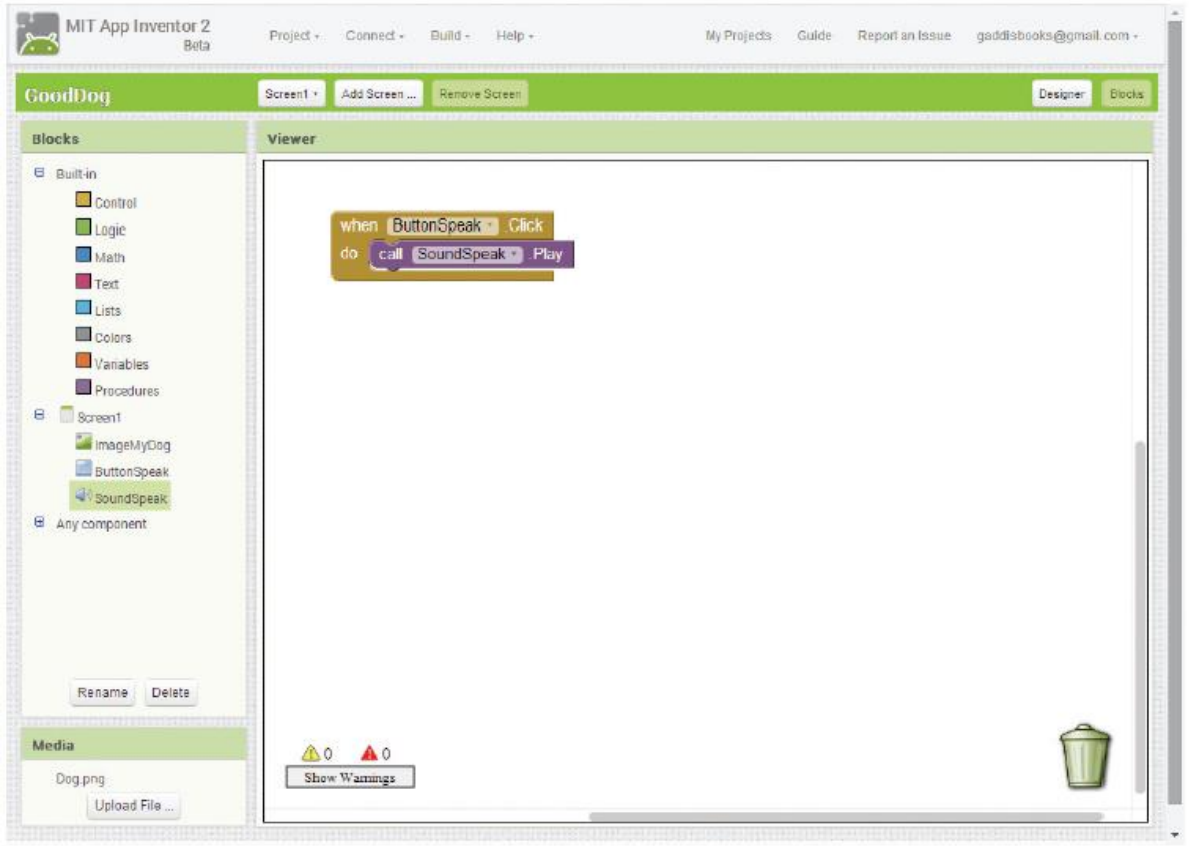
Figure 1-1 The App Inventor Designer (*Source:* MIT App Inventor 2, Pearson Education, Inc.)



(cc) BY-NC

Introduction

Figure 1-2 The Blocks Editor (*Source:* MIT App Inventor 2)

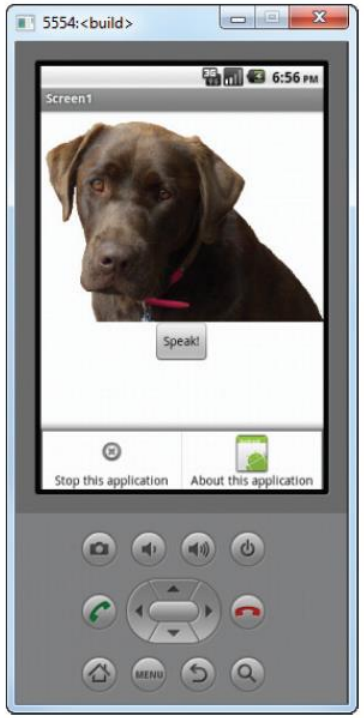


(cc) BY-NC

Introduction

- App Inventor provides an Android emulator that runs on your computer.
- The emulator (Figure 1-3) is a simulated Android phone.

Figure 1-3 The Android Emulator (*Source:* MIT App Inventor 2, Pearson Education, Inc.)



(cc) BY-NC

Introduction

- App Inventor Runs in the Cloud.
- App Inventor is part of MIT's Center for Mobile Learning.
- Advantages of the cloud-based approach
- You can access App Inventor from any computer connected to the Internet.
- Your files are maintained and backed up by the host.
- You can be sure you are always running the most recent version of App Inventor.

(cc) BY-NC

What Is a Computer Program?

- A computer program is a set of instructions that a computer follows to perform a task.
- A computer is a device that follows instructions for manipulation and storing data.
- When a computer is performing the instructions, we say it is *running* or *executing* the program.

(cc) BY-NC

What Is a Computer Program?

Algorithms and Programming Languages

- An algorithm is a set of well-defined, logical steps that must be taken in order to perform a task.
- The instructions have to be translated into machine language.
- In machine language, each instruction is represented by a binary number.
- A binary number is a number that has only ones and zeros. Here is an example.

```
1011010000000101
```

(cc) BY-NC

What Is a Computer Program?

Algorithms and Programming Languages

- Each language has its own syntax.
- Syntax is a set of rules that must be strictly followed.
- In traditional programming languages you convert your algorithm into a set of *statements*.
- Programmers call the statements *code.*
- An *executable program* is a file containing machine language instructions that can be directly executed by the computer.

(cc) BY-NC

What Is a Computer Program?

- Programming with App Inventor
- Beginning programmers frequently make typing mistakes resulting in *syntax errors*.
- In App Inventor, syntax errors never happen, because you do not type programming statements.
- Instead you drag and drop *code blocks*.
- The blocks can be “snapped” together like pieces of a puzzle.

(cc) BY-NC

Introducing App Inventor

- Each time you work with App Inventor you will perform the following steps:
  - Open a browser and go to the App Inventor website.
  - Either create a new project or open an existing project.
  - Open The Blocks Editor.
  - Connect either the Android emulator or an actual Android device.

(cc) BY-NC

Introducing App Inventor

The Designer

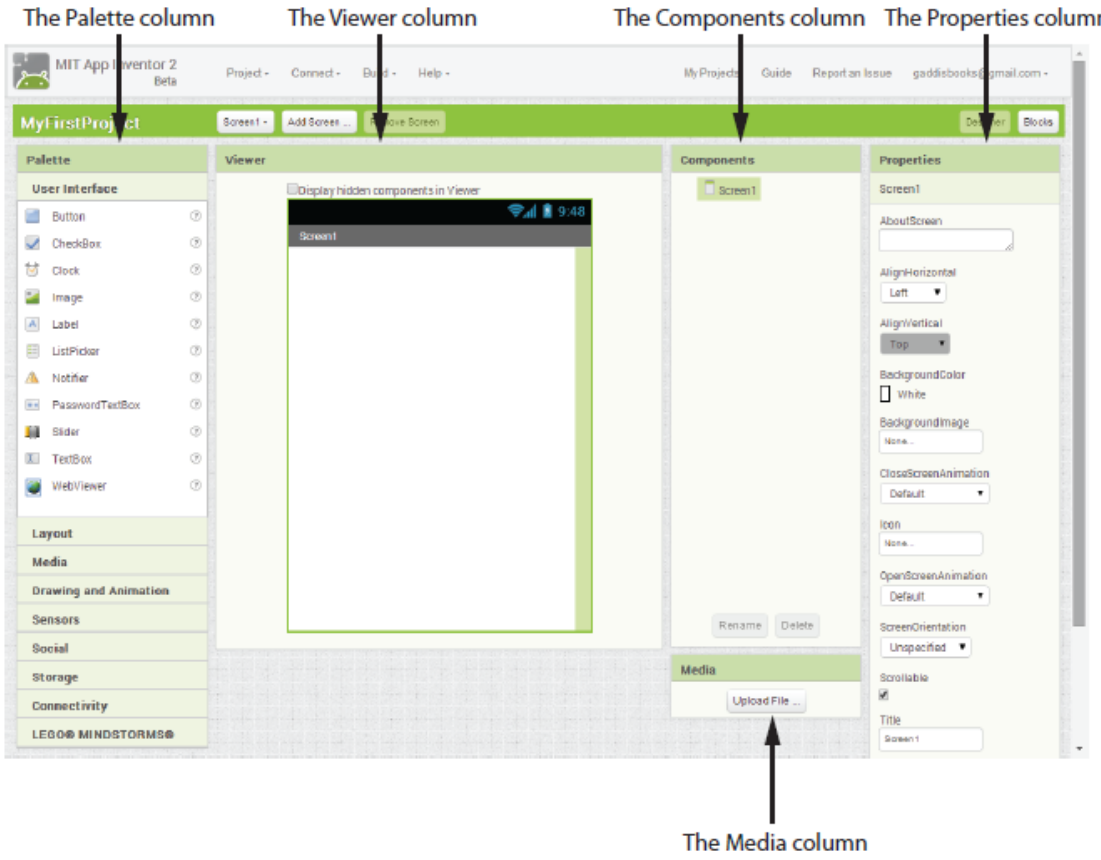
The Designer is organized into the following columns:

- The Pallet column.
- The Viewer column.
- The Components column.
- The Media column.
- The Properties column.

(cc) BY-NC

Introducing App Inventor

Figure 1-16 The Designer (*Source:* MIT App Inventor 2)



(cc) BY-NC

Introducing App Inventor

The Palette Column

- The Pallet provides a list of components.
- A *component* is an item that performs a specific purpose within an app.

(cc) BY-NC

Introducing App Inventor

- The different sections of the palette are:
  - ***User interface*** – The fundamental component for building an app's screen.
  - ***Layout*** – Provides components for organizing other components on the app's screen.
  - ***Media*** –
    - Provides components for taking photos.
    - Recording and playing videos.
    - Recording and playing sounds.
    - Picking Images.

(cc) BY-NC

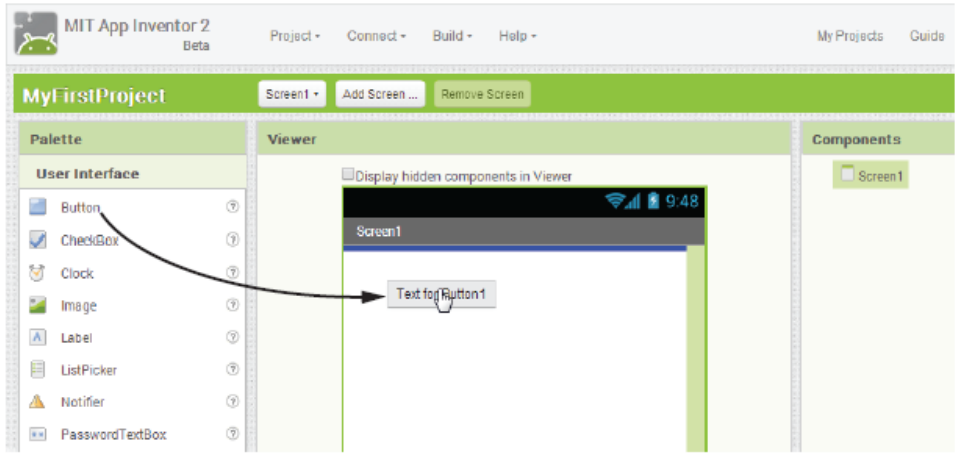
Introducing App Inventor

- The different sections of the palette are:
  - ***Drawing and Animation*** – Provides components for creating simple drawings and animations.
  - ***Sensors*** – Allows your app to access the device's accelerometer.
  - ***Social*** – Works with the phones contact list.
  - ***Storage*** – These components store data locally on a device or remotely on the Web server.
  - ***Connectivity*** – Provides components for launching external applications.

(cc) BY-NC

Introducing App Inventor

Figure 1-17 Creating a Component by Dragging it from the Palette to the Viewer (*Source:* MIT App Inventor 2)



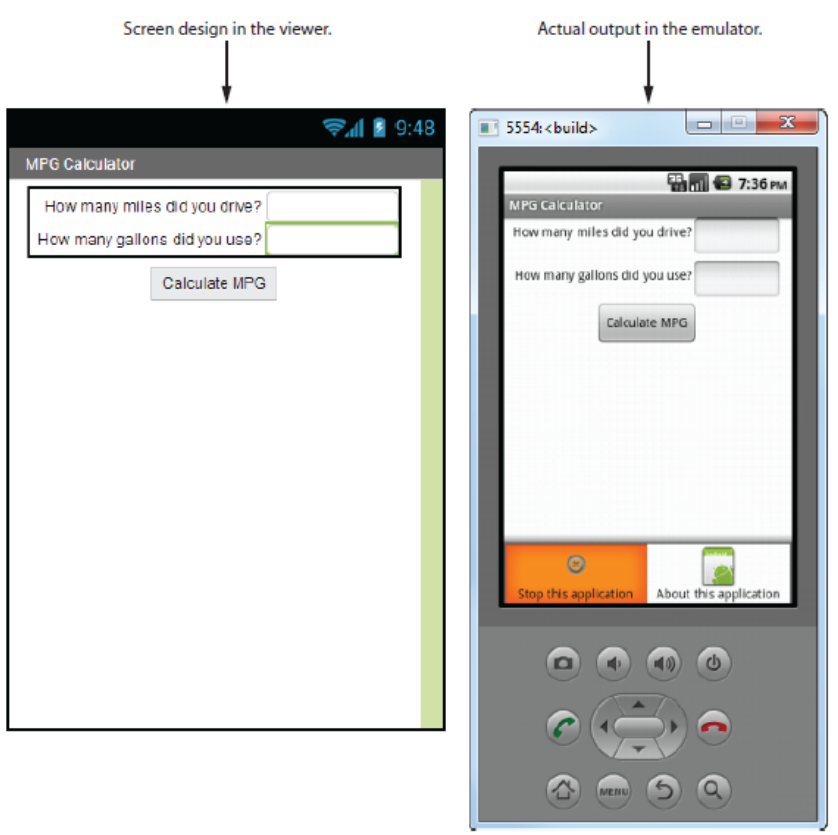
- The Viewer Column
  - You design an apps *user interface* by dragging components from the Pallet onto the simulated screen in the Viewer.
  - Components you place on the simulated screen in the Viewer might appear slightly different on the emulator screen.

(cc) BY-NC

Introducing App Inventor

- Notice the shapes of the text boxes and buttons are slightly different between the two screens.

Figure 1-18 A Screen in the Viewer and the Emulator (*Source:* MIT App Inventor 2)



(cc) BY-NC

Introducing App Inventor

The Components Column

Shows a hierarchical tree listing all of the components that you have placed your app.

The Media Column

Allows you to manage the media files (images, videos, and audio files).

(cc) BY-NC

Introducing App Inventor

- The Properties Column
- A Components appearance and other characteristics are determined here. Here are some examples:
- ***Label component*** – To display text on your devices screen.
- ***Image component*** – To display an image under the device's screen.
- ***Sound component*** – If you want the app to play a sound.

(cc) BY-NC

Introducing App Inventor

- Block's Editor
- A block is a shape that looks like a puzzle piece.

Figure 1-20 A Programming Statement Constructed from Code Blocks
(*Source:* MIT App Inventor 2)

(cc) BY-NC

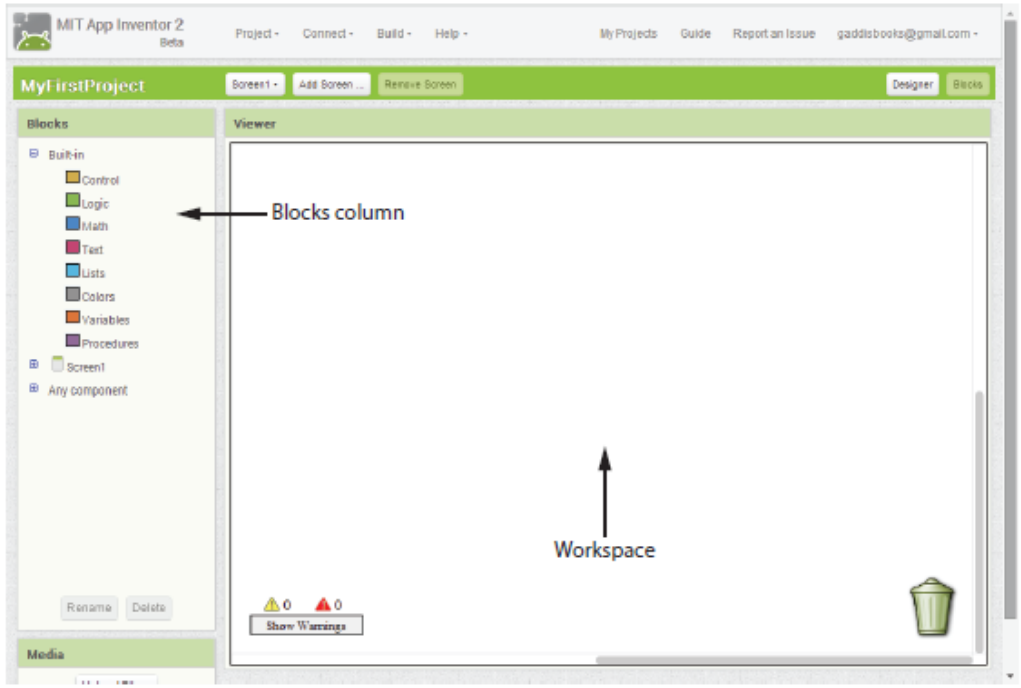
Introducing App Inventor

- The blocks column is organized in the following manner:
- ***Built-In*** – The basic blocks that make up the App Inventor language.
- ***Screen1*** – Each time you add a component to Screen1 in the Designer, a set of component blocks are added to the section.
- ***Any component*** – Allows a programmer to work with any component in the app.

(cc) BY-NC

Introducing App Inventor

Figure 1-21 The Blocks Editor (*Source:* MIT App Inventor 2)

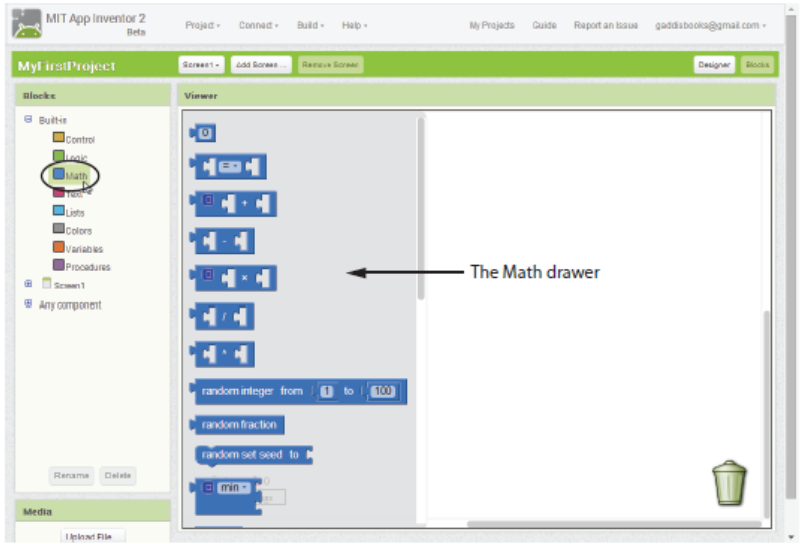


(cc) BY-NC

Introducing App Inventor

- The Built-in blocks
- Figure 1-23 Shows what happens when you click *Math*

Figure 1-23 The Math Drawer Opened (*Source:* MIT App Inventor 2)



(cc) BY-NC

Introduction

Figure 1-24 Top Part of the App Inventor Screen (*Source:* MIT App Inventor 2)

- The top part of the App Inventor screen shows the following items:
- ***Project*** – Start, save, and export projects.
- ***Connect*** – Connect to an Android device or the Android emulator.
- ***Build*** – Package an app so it can be shared.
- ***Help*** – Provides access to documentation, tutorials, and the App Inventor forum.

(cc) BY-NC

Introducing App Inventor

- ***My Projects*** – Displays a list of all the App Inventor projects that you have created.
- ***Guide*** – Opens a separate Web page containing the App Inventor documentation.
- ***Report an Issue*** – Takes you to the App Inventor support forum.

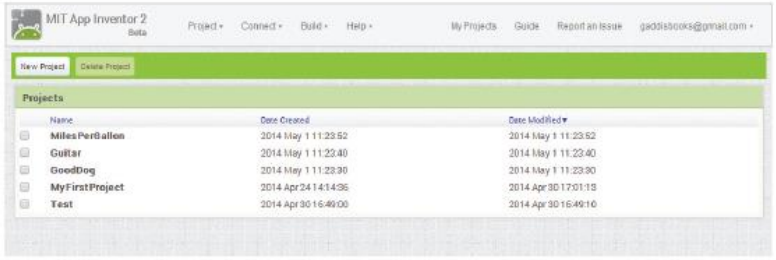
(cc) BY-NC

Getting Hands-On with App Inventor

Managing Projects

- Get to the My Projects screen by going to appinventor.mit.edu.
- Click the *Create* button.

Figure 1-26 The My Projects Screen (*Source:* MIT App Inventor 2)



- The *New* button creates a new project.
- The *Delete* button deletes the project.

(cc) BY-NC

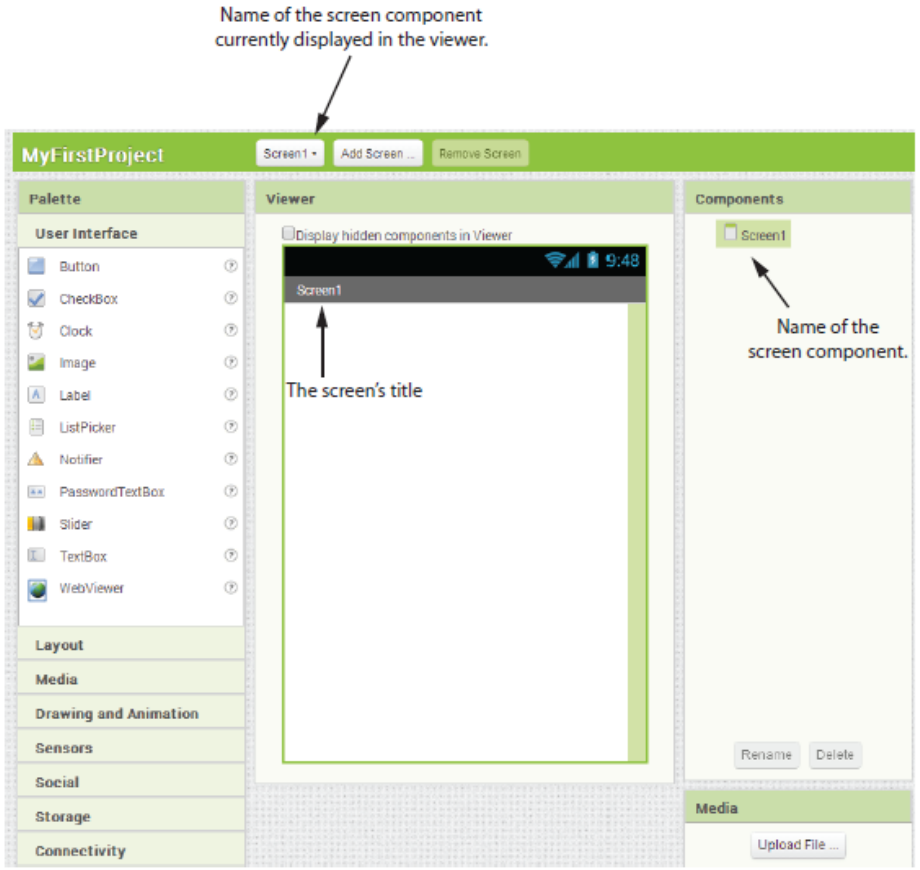
Getting Hands-On with App Inventor

- The App's `Screen1` Component
- A `Screen` is the most fundamental type of component.
- Each component in the app must have a unique name.
- By default an empty `Screen` component is named `Screen1`.
- You will want to change the default name to something meaningful.

(cc) BY-NC

Getting Hands-On with App Inventor

Figure 1-28 An App's Screen in the Viewer (*Source:* MIT App Inventor 2)



(cc) BY-NC

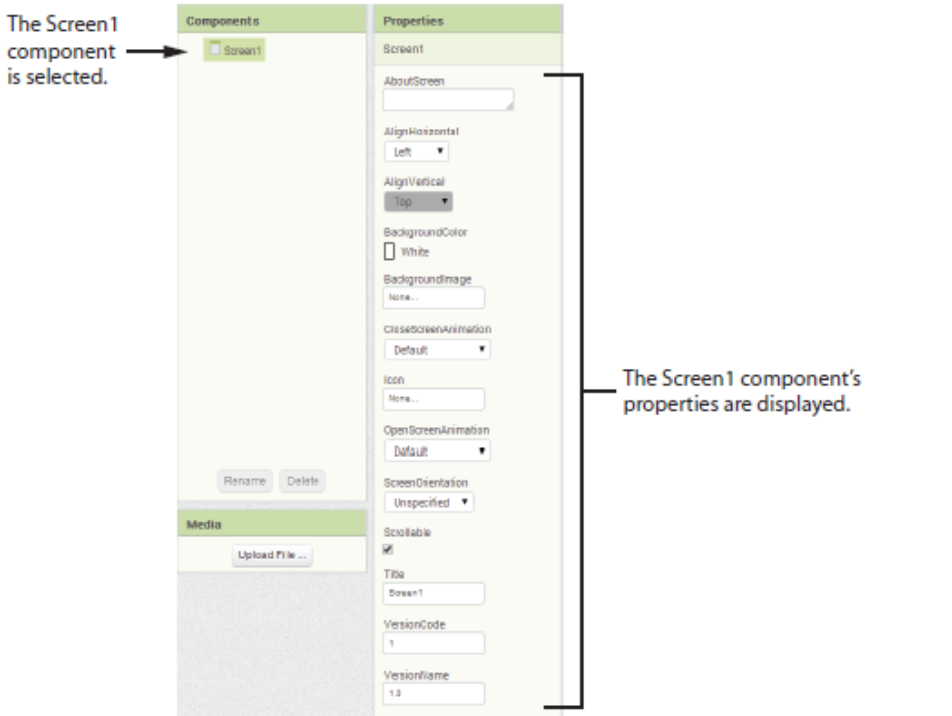
Getting Hands-On with App Inventor

- Working with the Properties Column
- Component properties are displayed in the Properties column.
- In Figure 1-29 notice the propertie's name is `Title`.
- Usually you will want to change the value of `Screen1` one complements the title property or something meaningful.

(cc) BY-NC

Getting Hands-On with App Inventor

Figure 1-29 The Properties Column, Showing the Selected Component's Properties (*Source:* MIT App Inventor 2)

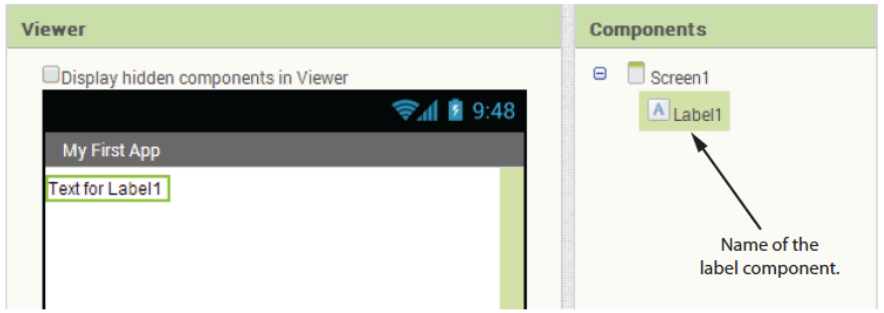


(cc) BY-NC

Getting Hands-On with App Inventor

- Label Components
- A `Label` compound it displays text on the app's screen.
- Create a `Label` component by dragging it from the User Interface section of the Pallet onto the app's screen in the Viewer.
- Figure 1-33 shows the Components column after a `Label` component has been created.

Figure 1-33 The Name of the Component Shown in the Components Column
(*Source:* MIT App Inventor 2)

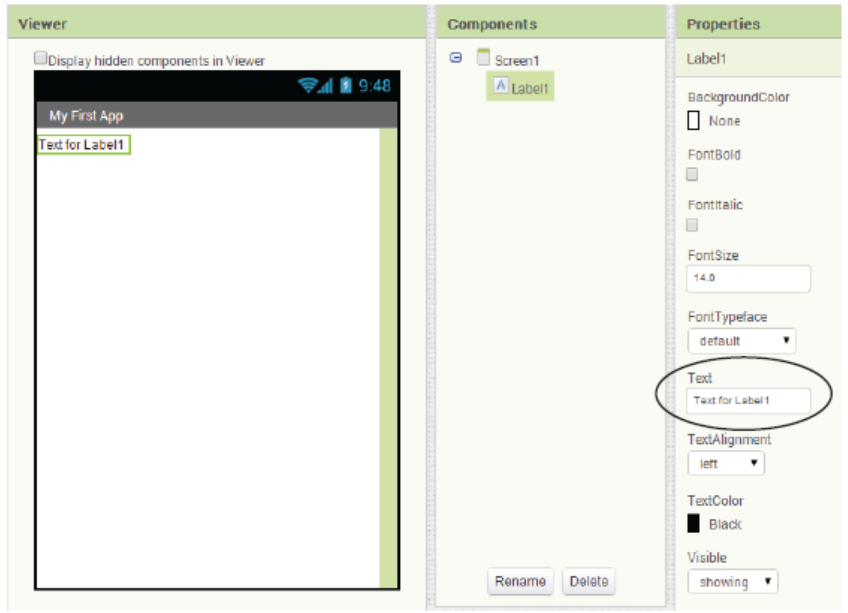


(cc) BY-NC

Getting Hands-On with App Inventor

- Once you have created a `Label` component, set it's text property to the text you want.

Figure 1-34 A Label Component's Text Property Determines the Text that the Component Displays (*Source:* MIT App Inventor 2)

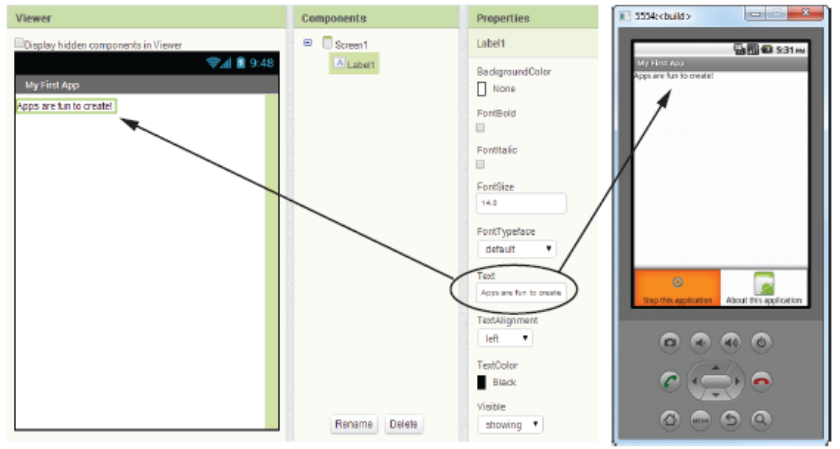


(cc) BY-NC

Getting Hands-On with App Inventor

- Figure 1-35 Shows an app with the `Label` component.
- The text property is set to *Apps are fun to create!*

Figure 1-35 A Label Component Displaying the Text *Apps are fun to create!*
(*Source:* MIT App Inventor 2)

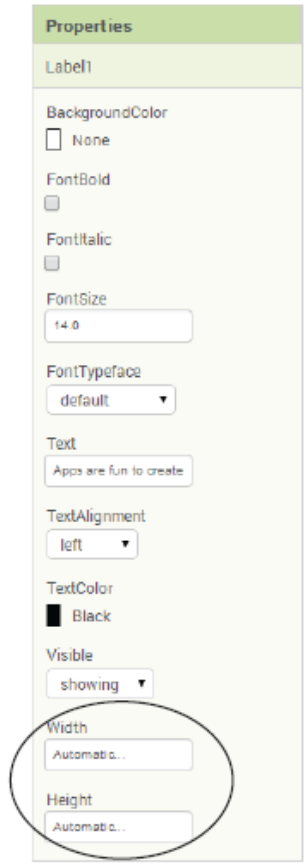


(cc) BY-NC

Getting Hands-On with App Inventor

- Label Width and Height

Figure 1-36 The Label Component's Width and Height Properties
(Source: MIT App Inventor 2)



(cc) BY-NC

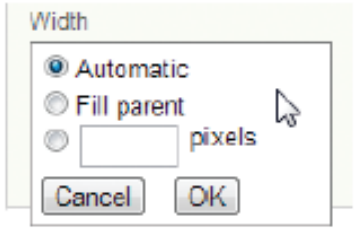
Getting Hands-On with App Inventor

- Property values you can set are `Width` and `Height`
- ***Automatic*** – The component's width will automatically adjust to accommodate the size of the label's text.
- ***Fill parent*** – The component will be as wide as the container.
- ***A specified number of pixels*** – You can specify a number of pixels for a components width and/or height. You should avoid this in most cases, because different devices have different screen sizes.

(cc) BY-NC

Getting Hands-On with App Inventor

Figure 1-37 Dialog Box to Set the Width Property (*Source:* MIT App Inventor 2)



(cc) BY-NC

Getting Hands-On with App Inventor

- Changing a Component's name
- When you create a component, App Inventor automatically gives it a name.
- Default names are not very descriptive.
- Change the component's name to something that is more meaningful.

(cc) BY-NC

Getting Hands-On with App Inventor

- Changing a Components Name
- In the Designer, you can use the Components column to change the name of any component (Except the `Screen1` component). Do so by:
- Clicking the name of the component in the Components column to select it.
- Click the *Rename* button at the bottom of the components column.
- The *Rename Component* dialog box shown in Figure 1-38 will appear.
- Enter the component's new name and click OK.

(cc) BY-NC

Getting Hands-On with App Inventor

Figure 1-38 Rename Component Dialog Box (*Source:* MIT App Inventor 2)

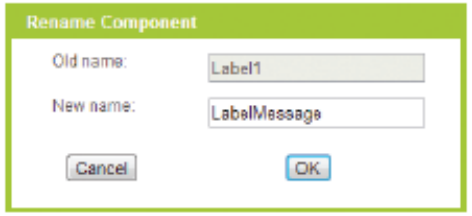
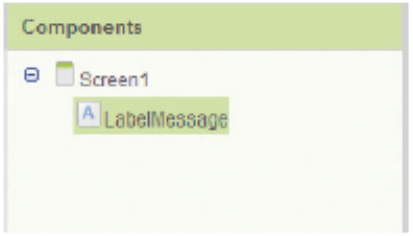


Figure 1-39 The Component's Name is Changed to `LabelMessage`
(*Source:* MIT App Inventor 2)



(cc) BY-NC

Getting Hands-On with App Inventor

Table 1-1 Legal and illegal component names (Source: Pearson Education, Inc.)

Name	Legal or Illegal?
`3rdTestScoreLabel`	Illegal because component names must start with a letter
`Label*Mobile*Number`	Illegal because the * character is not allowed. Component names can contain only letters, numbers, and underscores.
`Label Contact Name`	Illegal because component names cannot contain spaces
`Label_Contact_Name`	Legal

Rules and Conventions for Naming Components

- Component names can contain only letters, numbers, and underscores (_).
- The first character of the component name must be a letter.
- Component names cannot contain spaces.

(cc) BY-NC

Getting Hands-On with App Inventor

- Deleting Components
- To delete a component, click the *Delete* button that appears at the bottom of the Component's column.

(cc) BY-NC

Getting Hands-On with App Inventor

- `Button` Components
- Create a `Button` component by dragging it from the User Interface section of the Pallet to the app's screen in the Viewer.
- Once you create a `Button` component, you should change its name to something more descriptive.
- You should also change the component's `text` property to indicate what the button will do when it is clicked.

(cc) BY-NC

Getting Hands-On with App Inventor

Figure 1-41 A Button Component Displaying the Text *Click Me!* (Source: MIT App Inventor 2)

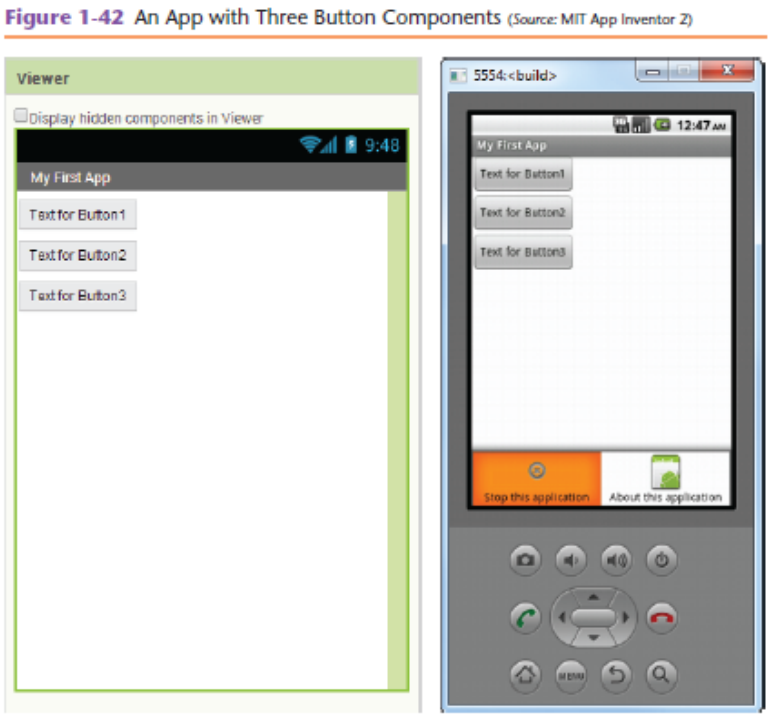
(cc) BY-NC

Getting Hands-On with App Inventor

•Screen Alignment

• Components are arranged vertically, from the top of the screen to the bottom of the screen.

• By default they are aligned along the left edge of the screen.



(cc) BY-NC John Robinson at Rowan University

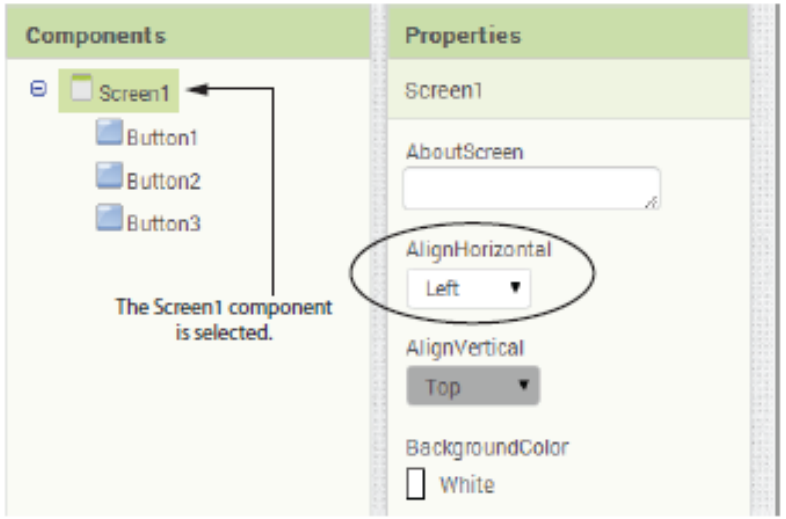
Getting Hands-On with App Inventor

- Screen alignment
- Screen components and have an `AlignHorizontal` Property.
- You can set the `AlignHorizontal` property to one of the following values:
- ***Left*** – along the left edge of the screen.
- ***Center*** – in the center of the screen.
- ***Right*** – Along the right edge of the screen.

(cc) BY-NC

Getting Hands-On with App Inventor

Figure 1-43 The AlignHorizontal Property *(Source: MIT App Inventor 2)*



(cc) BY-NC

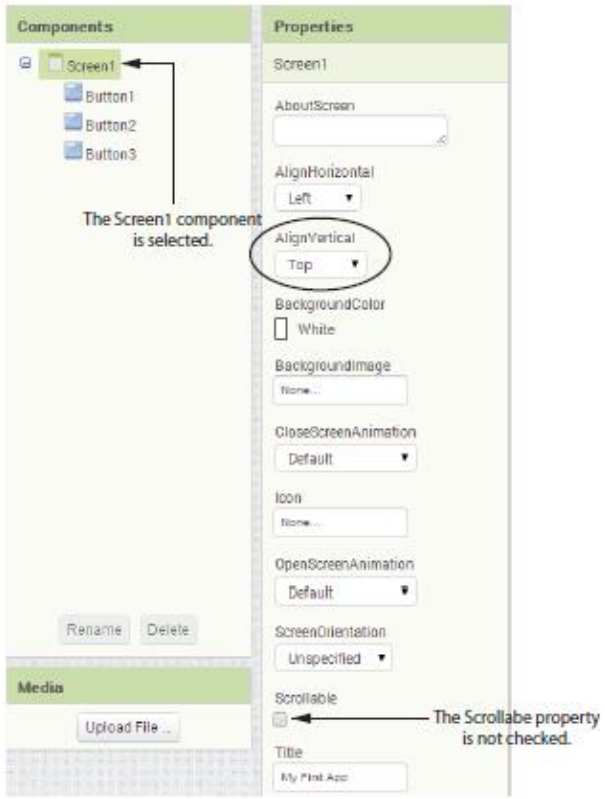
Getting Hands-On with App Inventor

- Screen Alignment
- Screen components also have an `AlignVertical` property.
- You can change the `AlignVertical` property only if the screen is not scrollable.
- You can set the `AlignVertical` property to one of the following values:
- ***Top*** – Along the top of the screen.
- ***Center*** – In the center of the screen.
- ***Bottom*** – Along the bottom of the screen.

(cc) BY-NC

Getting Hands-On with App Inventor

Figure 1-45 The AlignVertical Property (*Source:* MIT App Inventor 2)



(cc) BY-NC

Getting Hands-On with App Inventor

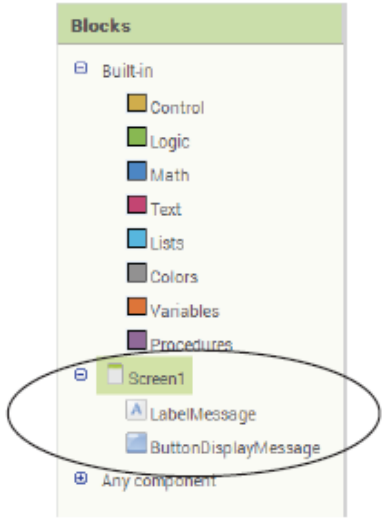
- Programming with Blocks
- Apps are *event-driven* programs.
- When an app is running, it waits for a specific event to happen.
- An event is an action such as the user clicking a button or sliding his or her finger across the device's screen.
- An incoming text message is also an event.
- An event also occurs when the user tilts or shakes the phone.

(cc) BY-NC

Getting Hands-On with App Inventor

- Programming with Blocks
- The Hello World app has a `Button` component named `ButtonDisplayMessage`.
- And a `Label` component named `LabelMessage`
- We need a block that executes when user clicks the `ButtonDisplayMessage` component.

Figure 1-59 The Component Entries in the Blocks Column *(Source: MIT App Inventor 2)*

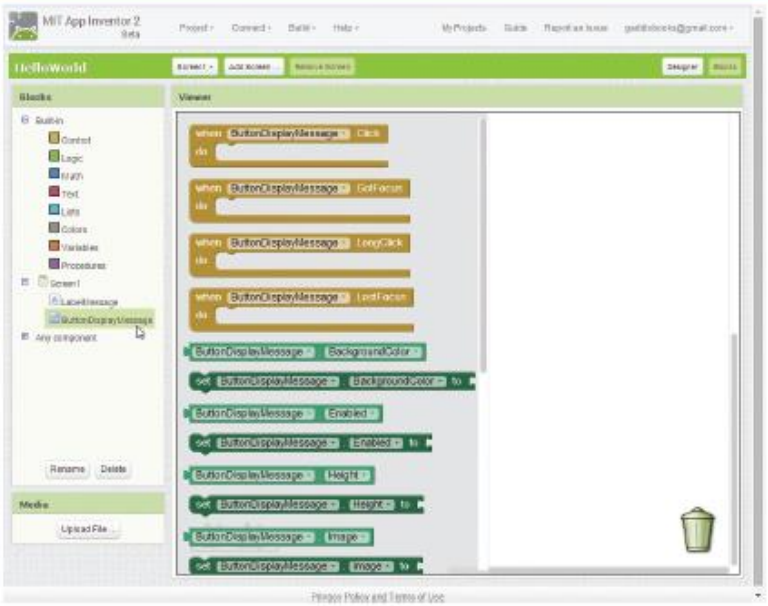


(cc) BY-NC

Getting Hands-On with App Inventor

- Programming with Blocks
- Because you want to create a block that executes when the `ButtonDisplayMessage` component is clicked, you need to click the `ButtonDisplayMessage` entry.

Figure 1-60 The `ButtonDisplayMessage` Drawer Open *(Source: MIT App Inventor 2)*



(cc) BY-NC

Getting Hands-On with App Inventor

Programming with Blocks

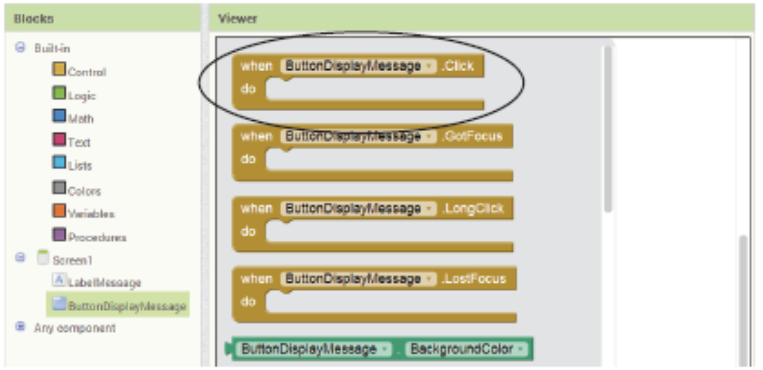
There are numerous blocks in this drawer. Here is a summary of the meaning of the colors.

- ***Brown blocks*** are called *event handlers*.
  - An *event handler* is a block that automatically executes when he specific event takes place.
- ***Light green blocks*** represent values that are related to the component.
- ***Dark green blocks*** perform actions with the component.

(cc) BY-NC

Getting Hands-On with App Inventor

Figure 1-61 The `when ButtonDisplayMessage.Click do` Block
(Source: MIT App Inventor 2)



Programming with Blocks

- Figure 1-61 Shows the topmost block inside the drawer. It reads:

```
when ButtonDisplayMessage.Click do
```

- This means when the `ButtonDisplayMessage` is clicked, *do this block*.

(cc) BY-NC

Getting Hands-On with App Inventor

Figure 1-63 You Complete the Block by Snapping Other Blocks into the Empty Space. (*Source:* MIT App Inventor 2)

Programming with Blocks

- The `ButtonDisplayMessage.Click do` block has an odd-shaped space in the middle.
- You can snap another block or set of blocks into this space.

(cc) BY-NC

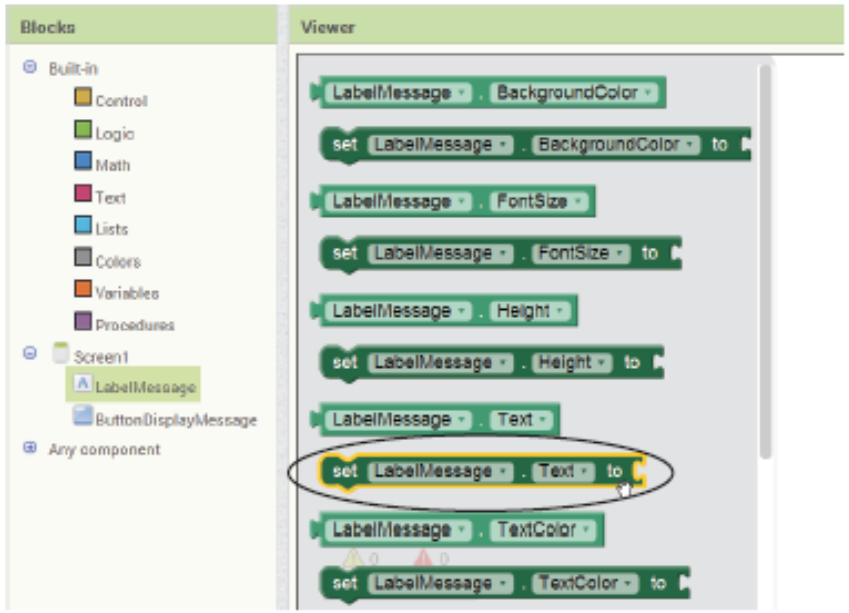
Getting Hands-On with App Inventor

- Programming with Blocks
- A `Label` component has a `Text` property.
- Find a block that sets the `LabelMessage` component's `Text` property.
- Snap it into the `when ButtonDisplayMessage.Click do` block.
- Open the drawer containing the blocks for the `LabelMessage` component. It reads:
  - `set LabelMessage.Text to`

(cc) BY-NC

Getting Hands-On with App Inventor

Figure 1-64 The `set LabelMessage.Text to` Block (*Source:* MIT App Inventor 2)



(cc) BY-NC

Getting Hands-On with App Inventor

Programming in Blocks

- To insert the block you click and drag it to the empty space inside the `when ButtonDisplayMessage.Click do` block.
- The `set LabelMessage.Text to` block is not a complete instruction.
- Noticed the opening on the right edge of the set `LabelMessage.Text to` block.
- You need to snap another block specifying a value into the socket.

(cc) BY-NC

Getting Hands-On with App Inventor

Figure 1-65 The `set LabelMessage.Text to` Block Inserted
(Source: MIT App Inventor 2)

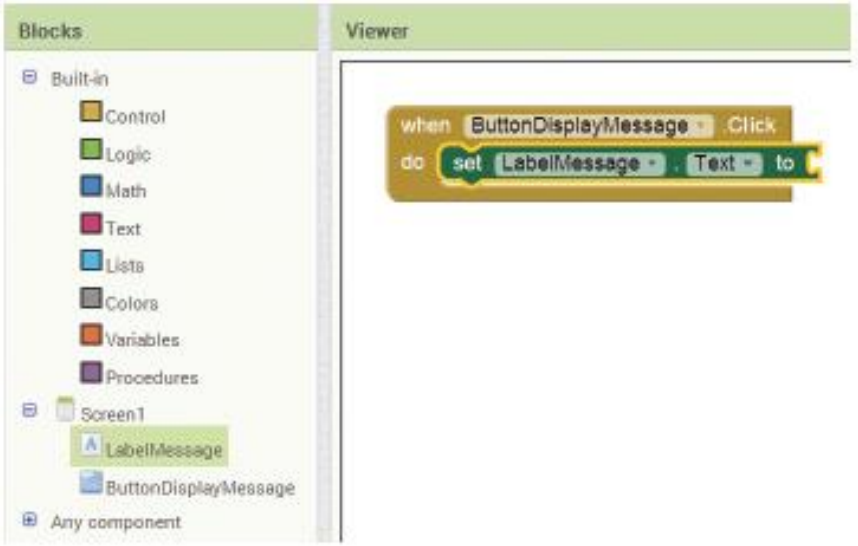


Figure 1-66 An Empty Socket (Source: MIT App Inventor 2)

(cc) BY-NC

Getting Hands-On with App Inventor

Programming in Blocks

- Click *Text* under *Built-in* in the Blocks column.
- A drawer will open as shown in Figure 1-67.
- In the figure is the *text string* block.
- Click the empty space, type *Hello World* then press enter.

(cc) BY-NC

Getting Hands-On with App Inventor

Figure 1-67 The Built-in `text string` Block (*Source:* MIT App Inventor 2)

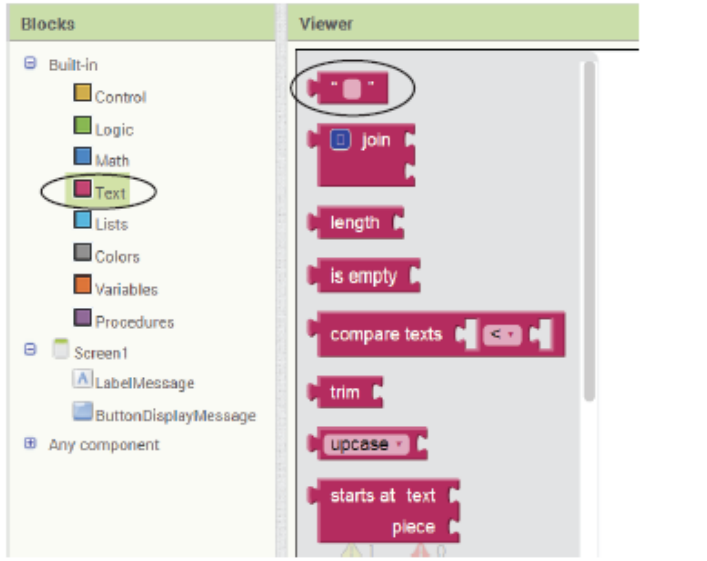
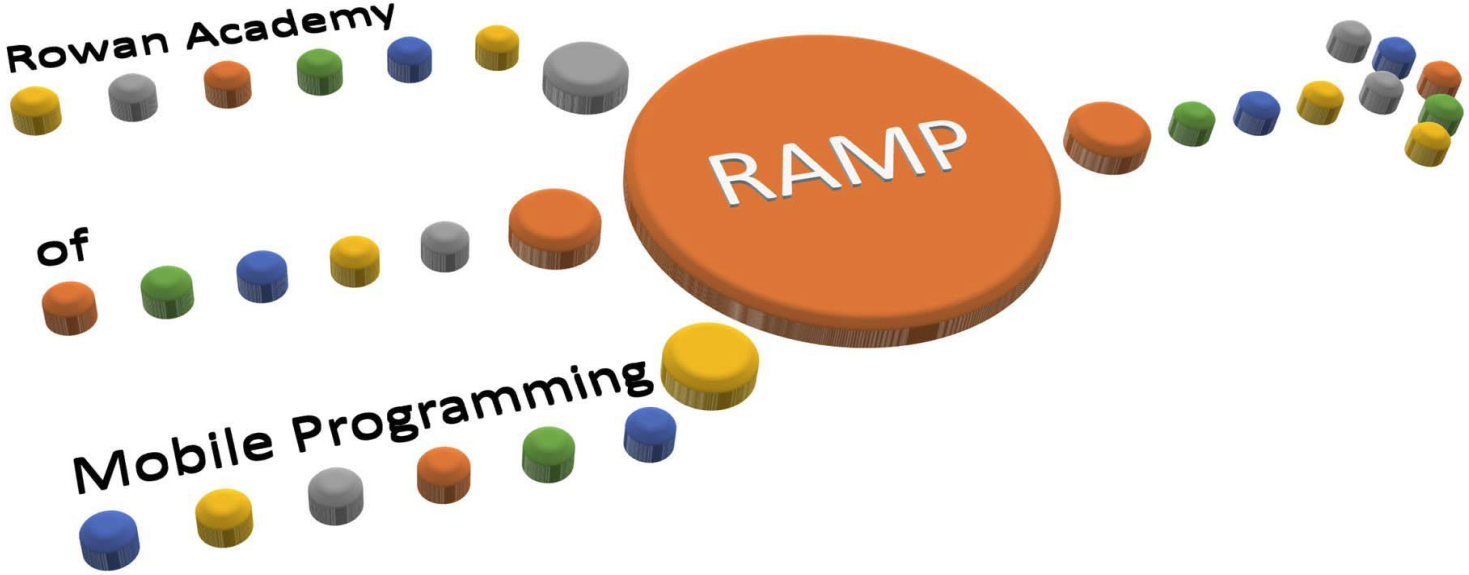


Figure 1-68 The Text String Block Snapped into the Socket of the `set LabelMessage.Text to` Block (*Source:* MIT App Inventor 2)

Figure 1-69 Changing the Value of the Text String Block to *Hello World* (*Source:* MIT App Inventor 2)

(cc) BY-NC



Mole Mash Game

(cc) BY-NC
John Robinson at Rowan University

Mole Mash Game Description

In our version of whack a mole, you will have thirty seconds to try and get the highest score you can. If your total score after the thirty seconds is high enough, you will be able to collect bonus points by shaking your device! Final Scores are then displayed on another we will call the ScoreScreen. On the ScoreScreen page, the high score is displayed along with the score from the previous game.

(cc) BY-NC

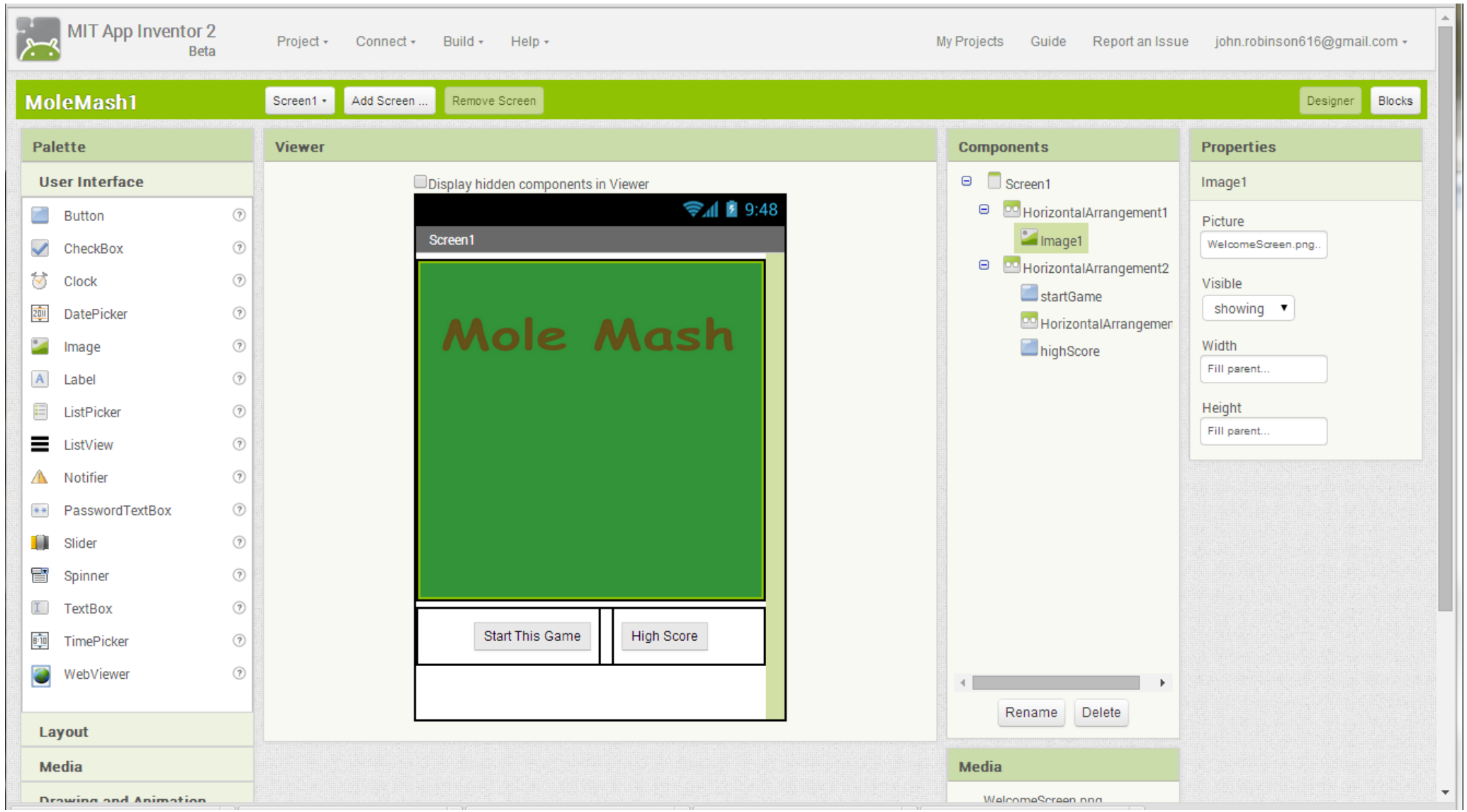
Android & Computer Science Concepts Covered in Mole Mash Game

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

- The Android accelerometer sensor
- Android event handling
- How conditional and control statements are used
- Data structures and Abstractions
- Parameter passing and Data Storage
- High level languages translated into low level languages.

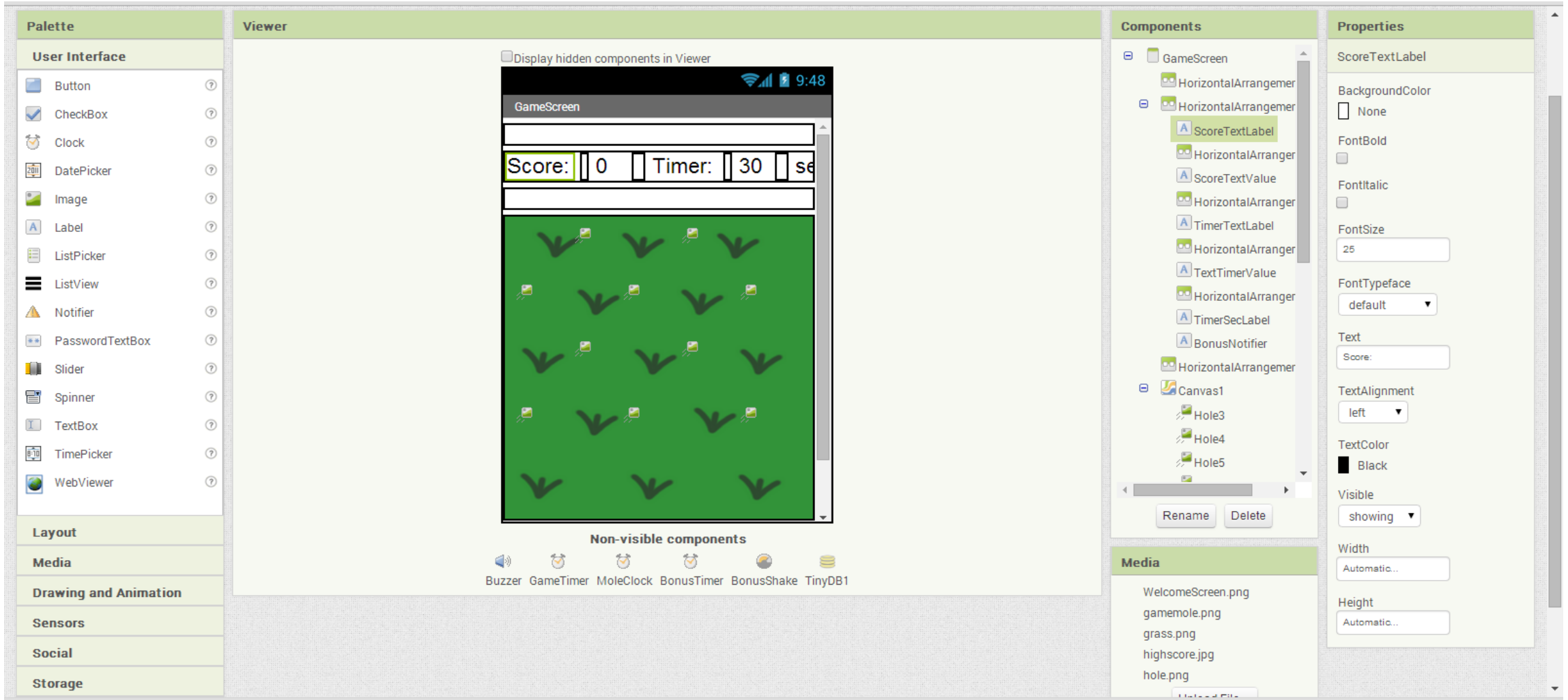
(cc) BY-NC John Robinson at Rowan University

Screen 1



(cc) BY-NC John Robinson at Rowan University

Game Screen top

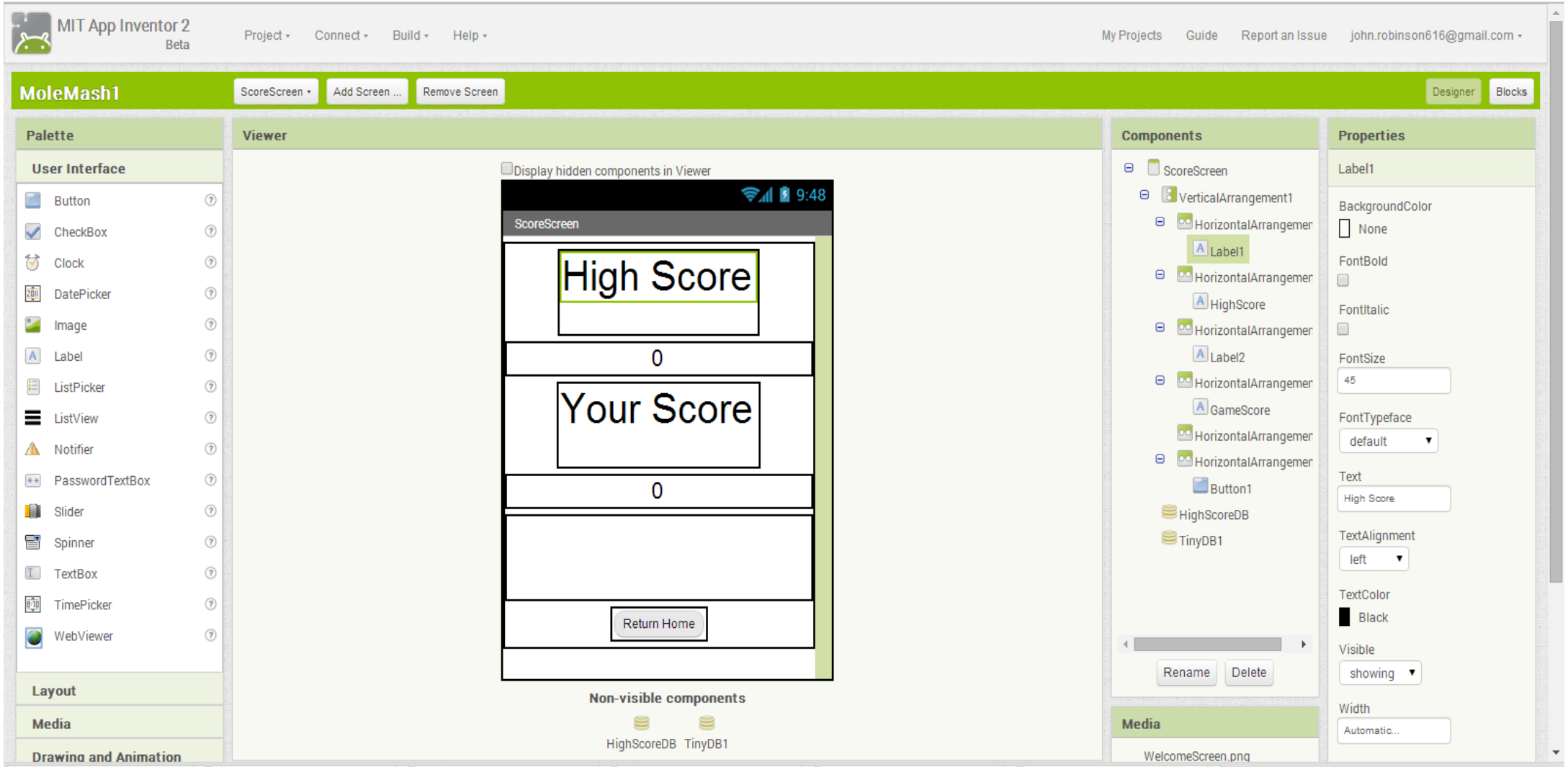


(cc) BY-NC John Robinson at Rowan University

Game Screen bottom

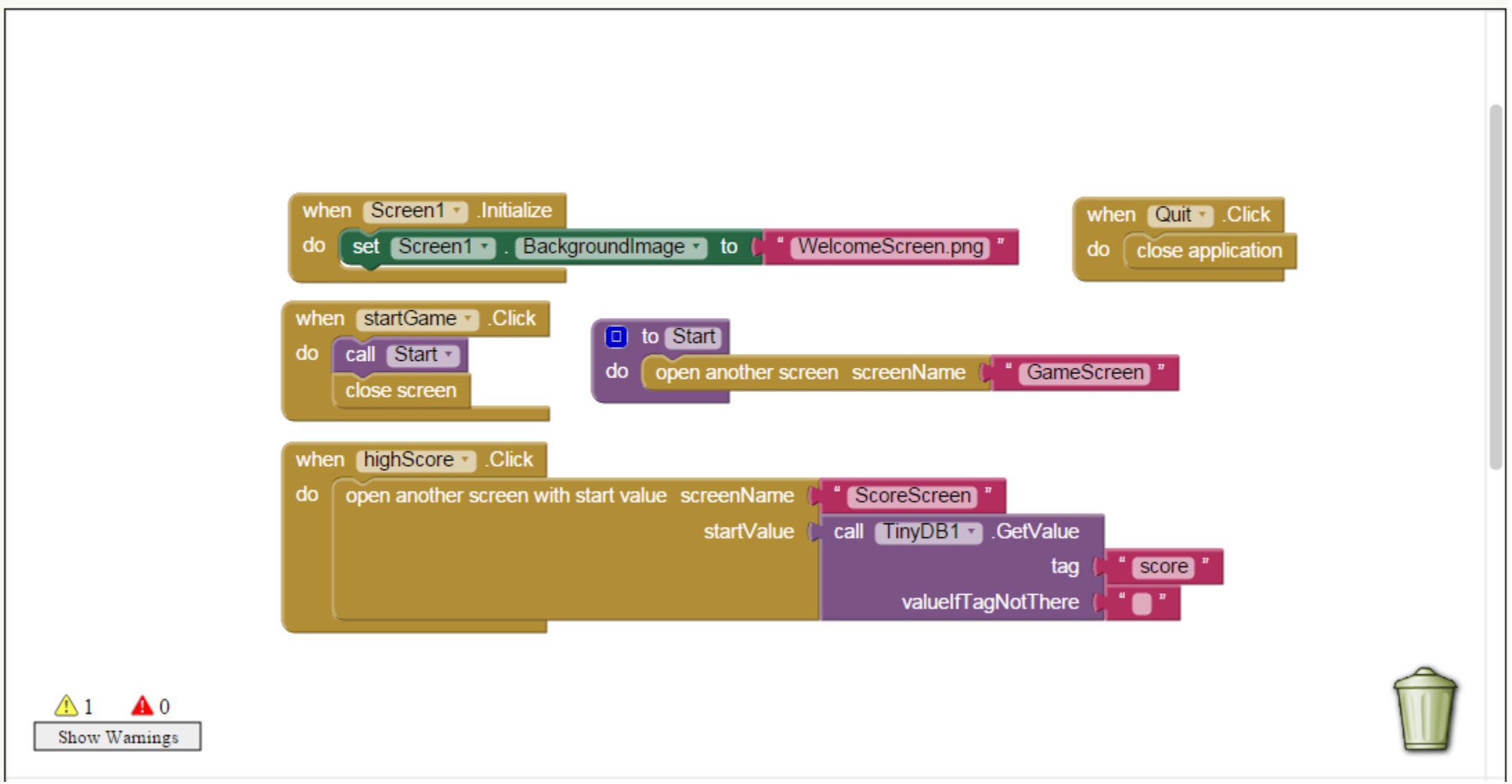
(cc) BY-NC John Robinson at Rowan University

Score Screen

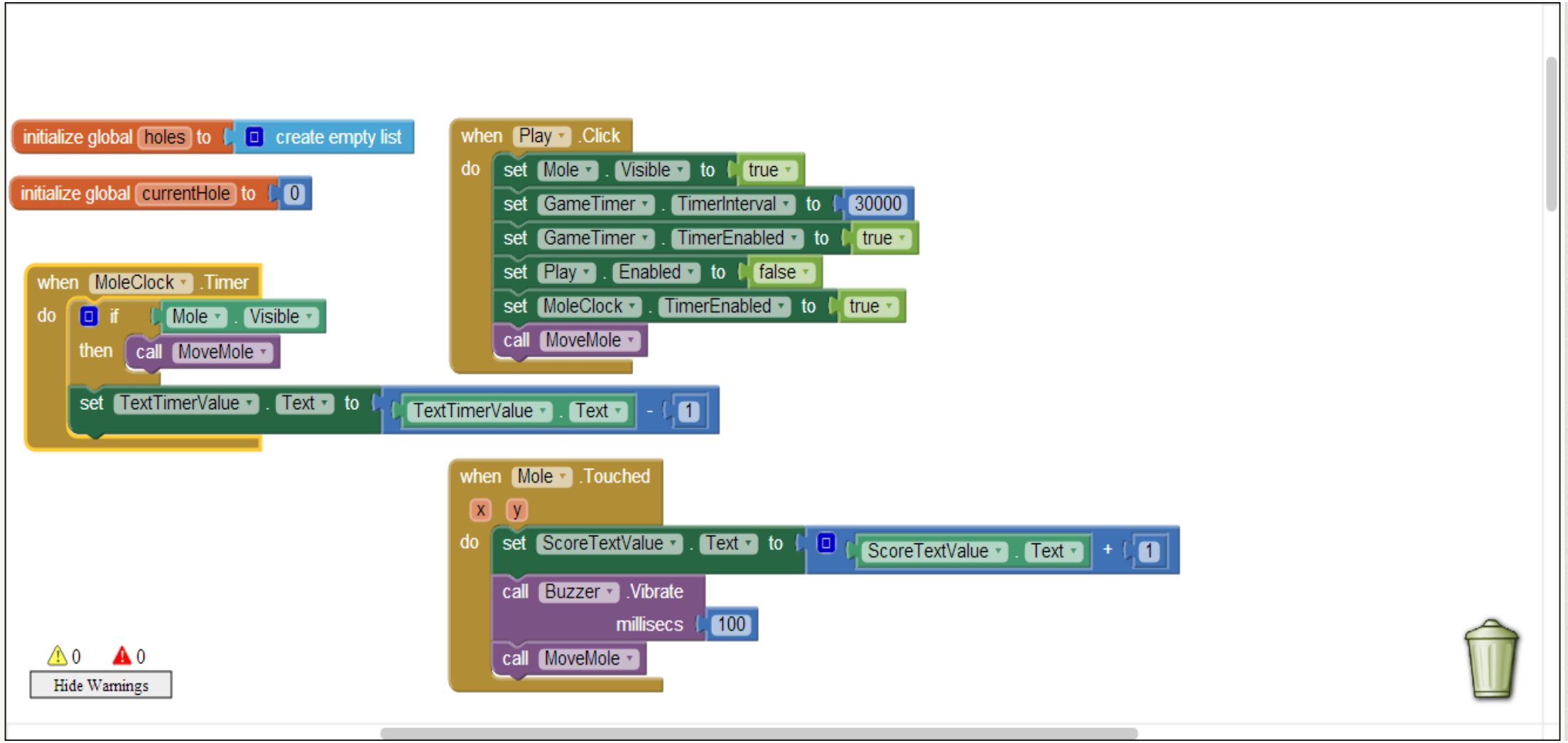


(cc) BY-NC John Robinson at Rowan University

Screen 1 Logic

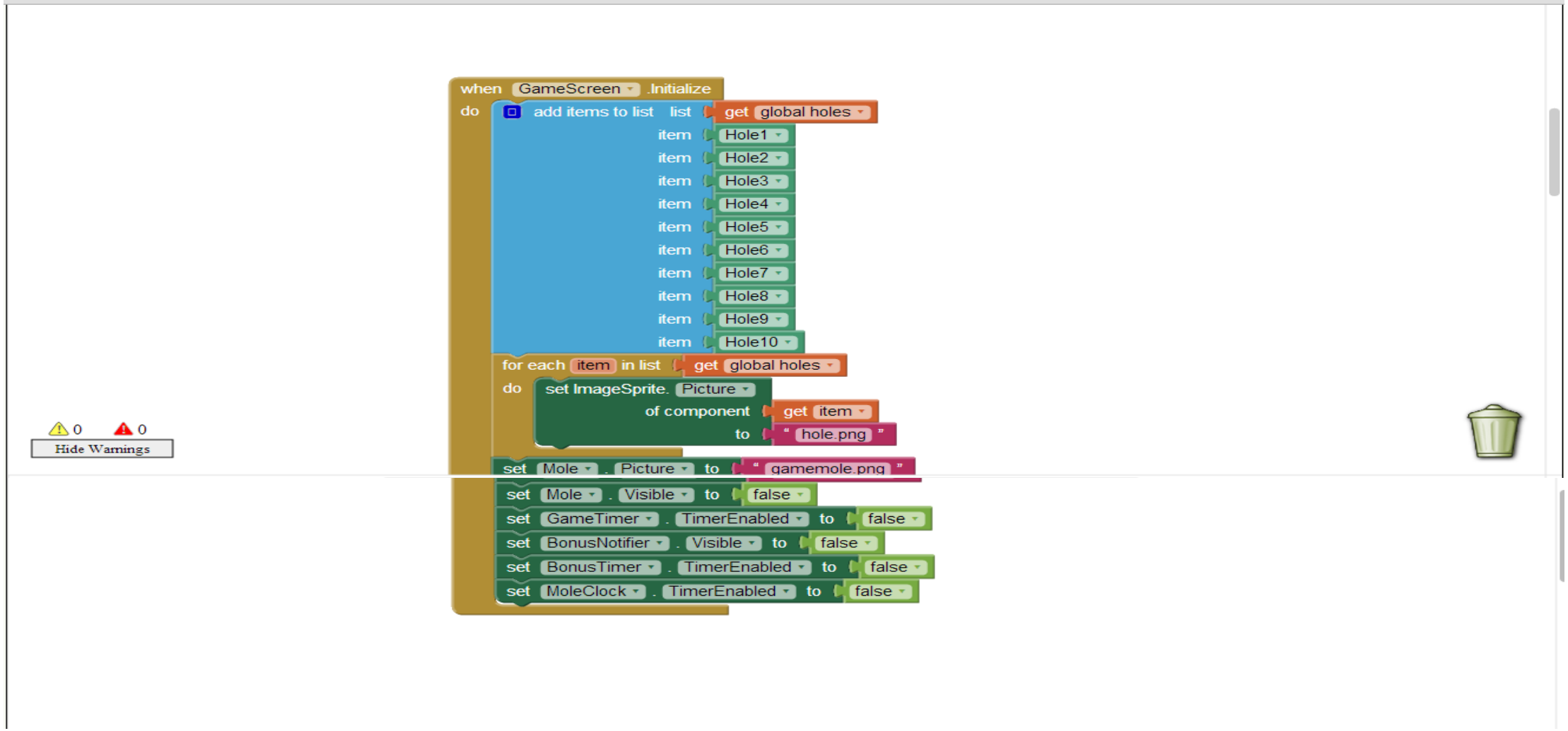


Game Screen Logic



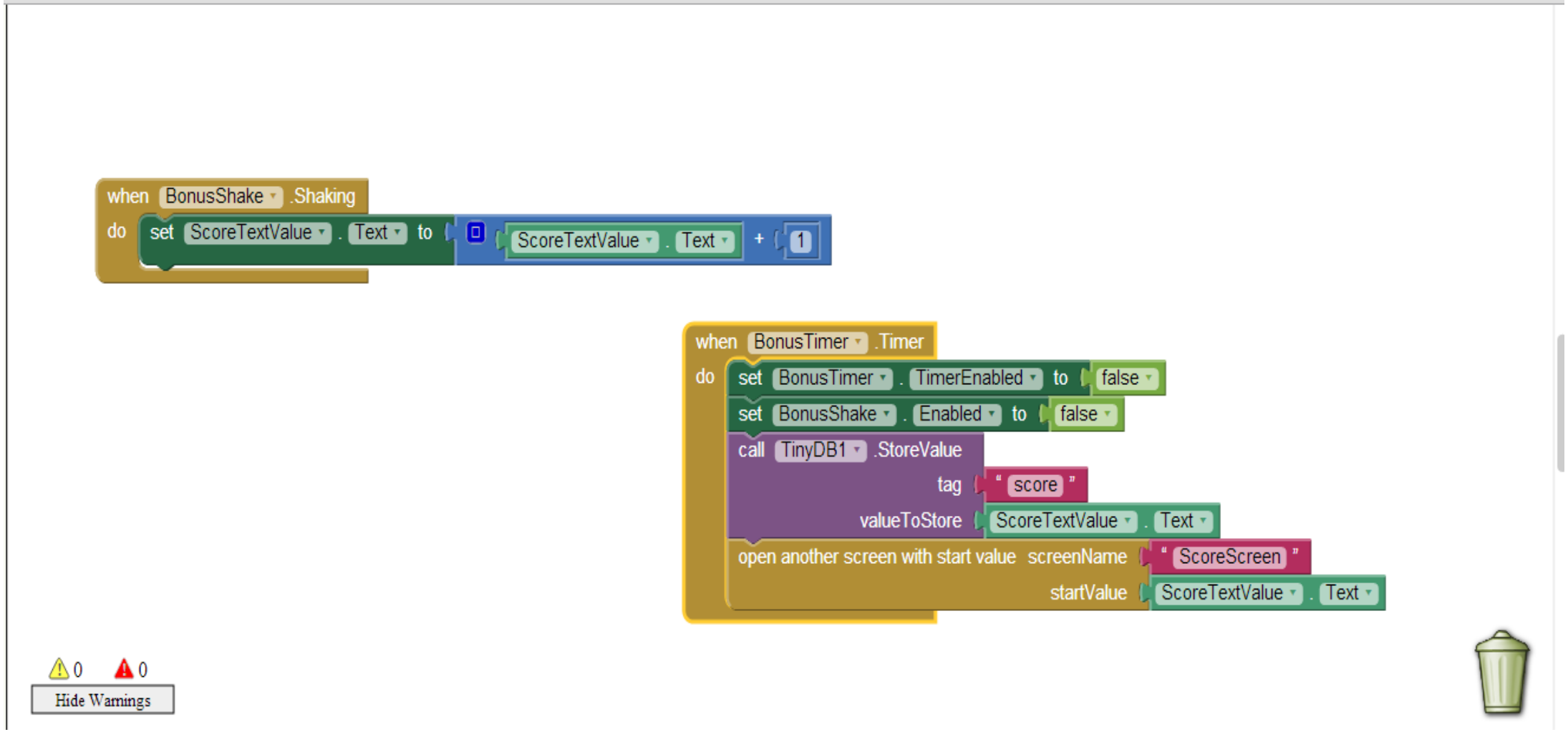
(cc) BY-NC John Robinson at Rowan University

Game Screen Logic



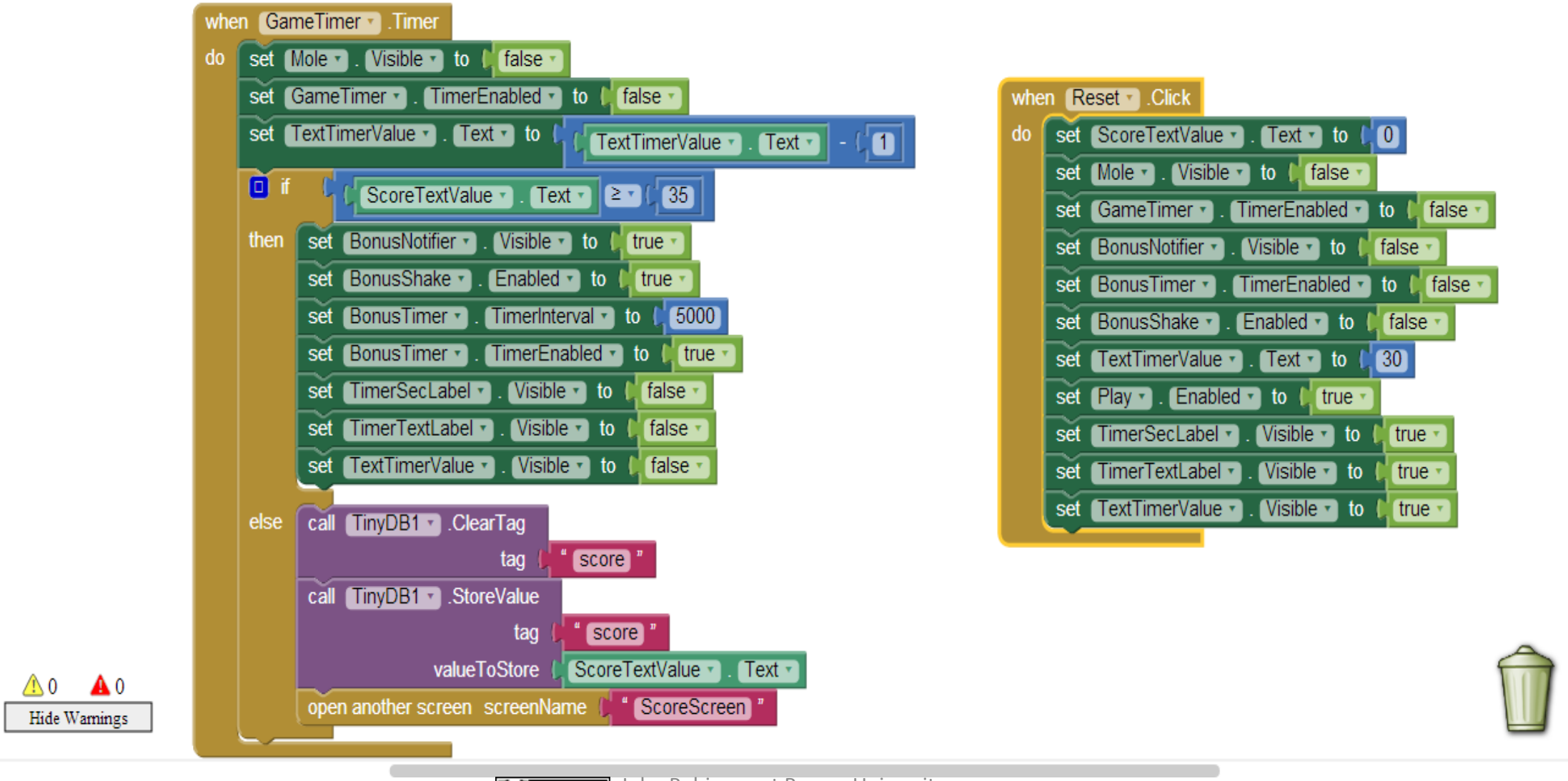
(cc) BY-NC John Robinson at Rowan University

Game Screen Logic



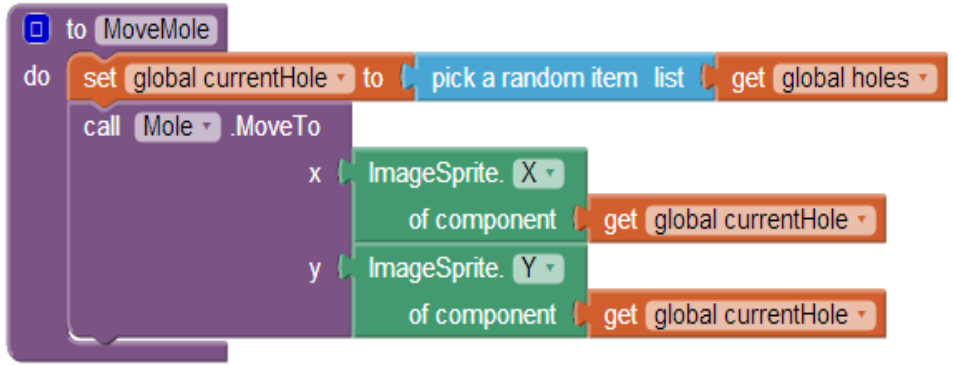
(cc) BY-NC John Robinson at Rowan University

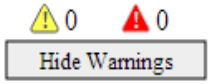
Game Screen Logic



(cc) BY-NC John Robinson at Rowan University

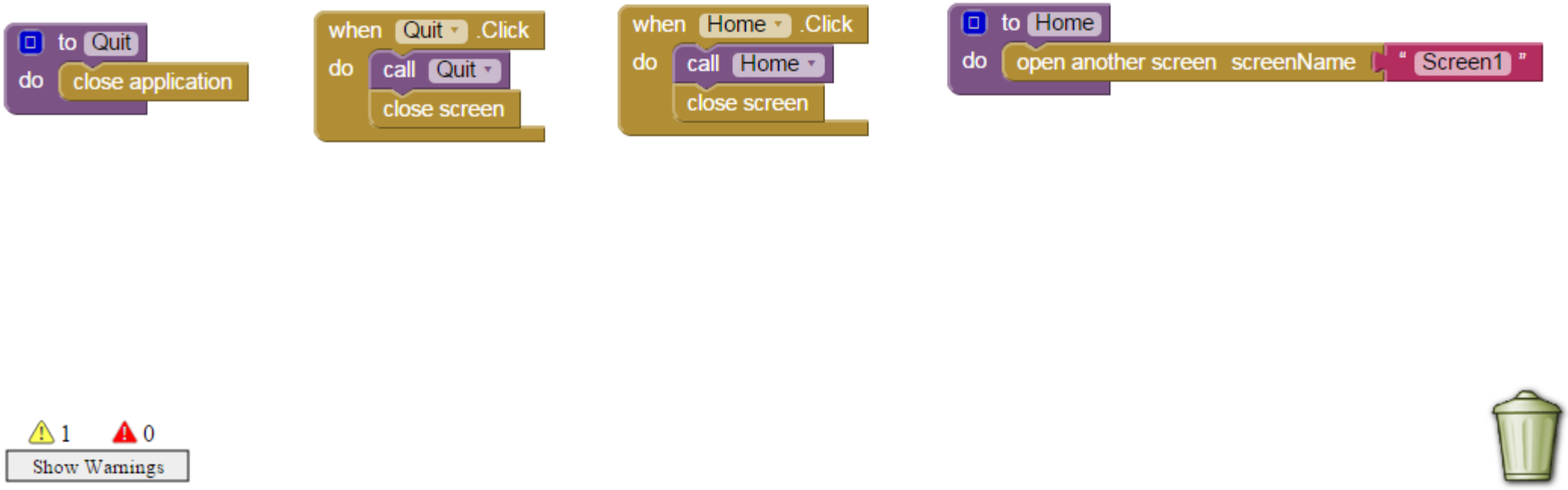
Game Screen Logic



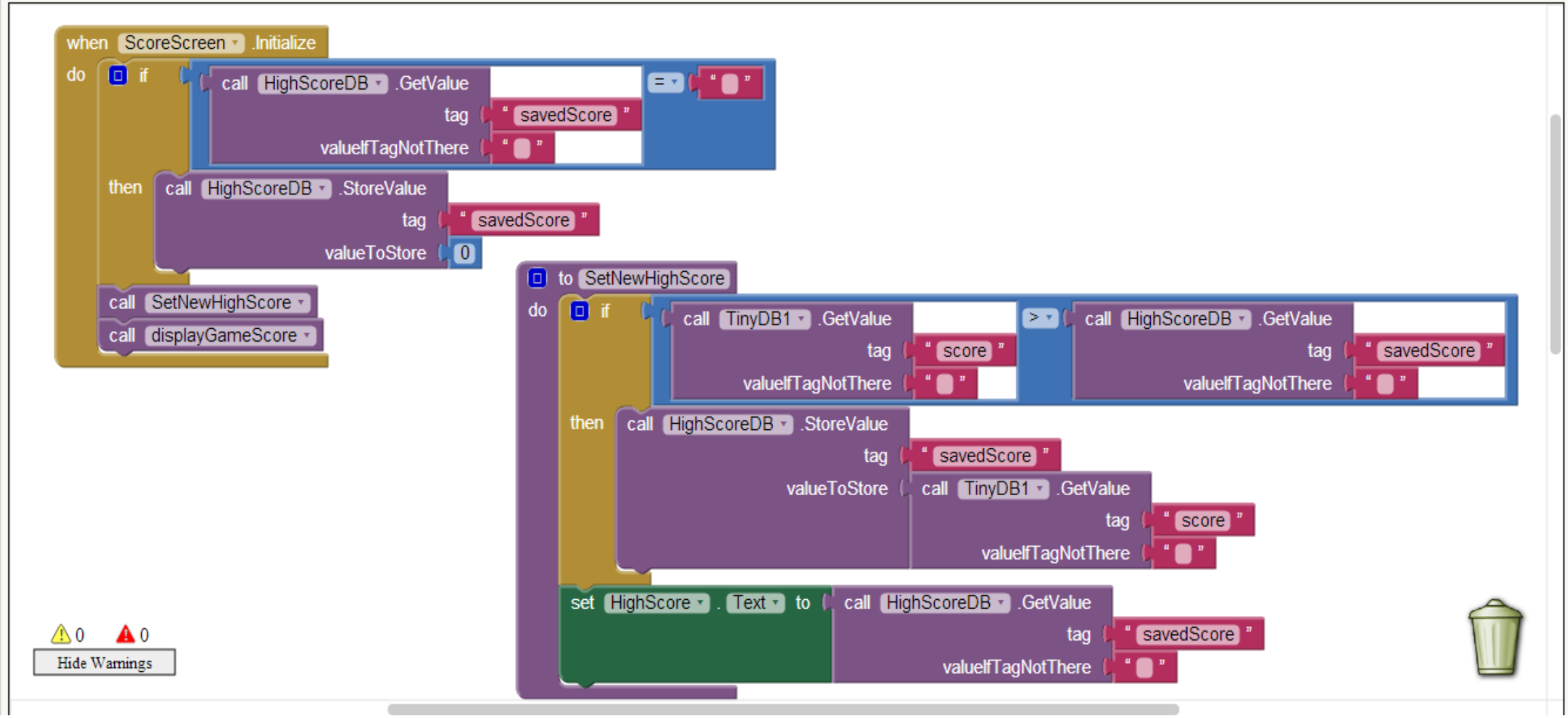


(cc) BY-NC

Game Screen Logic



Score Screen Logic



(cc) BY-NC John Robinson at Rowan University

Score Screen Logic