CoolThink@JC -
Computational Thinking Education Programme

Course Code & Title

CF3E Computational Thinking Development - Level 3

Unit Code & Topic

L3U8.6-8.8
Sketch and Guess

香港教育大學
The Education University of Hong Kong

Massachusetts Institute of Technology

Funded by
香港賽馬會慈善信託基金
The Hong Kong Jockey Club Charities Trust
同心同步同進 RIDING HIGH TOGETHER

These teaching and learning material were co-created by The Education University of Hong Kong, and Massachusetts Institute of Technology, and funded by The Hong Kong Jockey Club Charities Trust. (Revised January 2021)

Table of Contents

1. Synopsis

In this six-lesson unit, students will create a complex multiplayer drawing game, Sketch and Guess, in which a player can draw pictures and the other player guesses the drawing, similar to the game of Pictionary. In the first lesson, students will play with the finished app and determine roles for the Sketcher and Guesser. In Lesson 1, students gain an understanding of an app with three screens. The focus of Lesson 1 is algorithmic thinking. In the second lesson, students will add drawing capability to the app. The focus for Lesson 2 is also algorithmic thinking. In the third lesson, students will add the ability to "send" drawings to other app users, so that they can watch the user draw in real time. The focus for Lesson 3 is data structures. In the fourth lesson, through the unplugged activity, students deepen their understanding of CloudDB and code the functionality of the GuesserScreen in the app. The focus for Lesson 4 is data structures. In the fifth lesson, students code the passing of information about what is being drawn by the Sketcher. The focus for Lesson 5 is Naming and Variables. In the final lesson, students add the capability to check for a correct answer when guessing. The focus for Lesson 6 is conditionals. Teachers may encourage students to add various features (such as multiple colours and changing line width) with the help of the *Student Guide: Challenge*, making the app more user-friendly and fun. Students will apply the computational thinking practice of being incremental and iterative, since they will modify and add complexity to the app over six lessons. As always, testing and debugging will be necessary to make sure the apps work correctly.

2.Unit Intended Learning Outcomes (UILOs)

After completing this unit, students will be able to:

UILO$_1$: make a multiplayer drawing app that uses CloudDB;

UILO$_2$: apply the computational thinking concepts of sequences, events, conditionals, parallelism, naming and variables, operators, and data structures in creating an app;

UILO$_3$: demonstrate understanding of how to use CloudDB to pass multiple pieces of information between devices;

UILO$_4$: apply computational thinking practices of being incremental and iterative, and testing and debugging;

UILO$_5$: use the computational thinking practice of algorithmic thinking to code a multiscreen drawing app;

UILO$_6$: use the computational thinking practice of abstracting and modularizing to decompose a complex app into parts;

UILO$_7$: develop their computational identity and confidence by developing a complex multiplayer game; and

UILO$_8$: work collaboratively to code and test a working multiplayer app.

3. Mapping with the Computational Thinking Framework

The following tables show the alignment of this unit with the intended learning outcomes of the computational thinking framework. The entries indicate the expected relevance of this unit to each outcome:

✓✓✓	:	High relevance
✓✓	:	Some relevance
✓	:	Low relevance

Computational Thinking Concepts

L3U8.6-8.8: Sketch and Guess		
1. Sequences	✓✓	Correct sequences are important in the StartButton.Click event for the SketcherScreen.
2. Events	✓✓✓	Events used include Canvas.Dragged, CloudDB.DataChanged, Screen.Initialize, Spinner.AfterSelection.
3. Repetition		
4. Conditionals	Lesson 6 focus	Complex conditionals are required, such as nested if-statements.
5. Parallelism	✓✓	Multiple students may play at one time, so parallelism occurs.
6. Naming and Variables	Lesson 5 focus	Descriptive naming of variables, components, and tags in a complex app are necessary.
7. Operators	✓✓	Logic operators “and” and “not” are required to make the app function correctly.
8. Data Structures	Lessons 3, 4 focus	This app requires list manipulation and CloudDB.

Computational Thinking Practices

L3U8.6-8.8: Sketch and Guess		
1. Reusing and remixing		
2. Being incremental and iterative	✓✓✓	Students will build the app incrementally over the six class periods.
3. Abstracting and modularizing	✓✓✓	Students decompose the complex app into parts, and implement each part one at a time.
4. Testing and debugging	✓✓✓	Students test over multiple devices with a partner.
5. Algorithmic thinking	Lesson 1, 2 focus	Algorithmic thinking is required to understand the complex functionality of drawing and sharing data between devices.

Computational Thinking Perspectives

L3U8.6-8.8: Sketch and Guess		
1. Expressing	✓✓✓	Students can express themselves by building a drawing app.
2. Connecting	✓✓✓	Students connect with their real life by making a game they can play with friends.
3. Questioning	✓✓✓	Students will face challenges learning how to coordinate multiple players.
4. Computational identity	✓✓✓	Students build their confidence by building a complex game.
5. Digital empowerment	✓✓✓	Students feel empowered by building an app that can be played by many others.

4. Learning Prerequisites

Students should have completed Level 2 and Unit 1-5 of Level 3 of the CoolThink@JC Computational Thinking Education Programme curriculum.

5. Pedagogical Approach

The suggested pedagogical approaches of this unit are as follows:

- Class demonstration;
- Pair Programming;
- Unplugged Activities; and
- Testing and debugging.

6. Lesson Plan

This unit consists of 6 lessons of 35 minutes, with post-lesson activities after each period.

Lesson 1

Time	Activity
10 mins	**Introduction to Drawing with Canvas Component** Demonstrate the completed app (SAG_MultiScreen_complete.aia) and let students try it.
5 mins	**Review and Introduction to Lesson** 1. Review the functions for the Sketcher and the Guesser in the app. 2. Students complete Page 1 of the *Student Guide: Lesson 1*.
20 mins	**Decomposition of Parts of App** 1. Ask students if they think this is a complex app to build. 2. Explain that it helps to break down the parts of a big app into smaller parts, and then build each part, one at a time. 3. Ask students to complete the *Student Guide: Lesson 1*. (1) List out the tasks and abilities of each player in the game. (2) Fill out a sequencing diagram to decompose. (3) Test the opening and closing of screens in the project template.
Post-Lesson Activities	(Required) Complete the *Student Guide: Lesson 1* if not completed in class.

Lesson 2

Time	Activity
10 mins	**Introduction to Drawing with Canvas Component** Introduce drawing on a Canvas and how lines are drawn based on user input.
20 mins	**Coding of Simple Drawing App** 1. Ask students to pair up and work using the Pair Programming model to build a drawing app with the help of the *Student Guide: Lesson 2*. 2. Test and debug the app.
5 mins	**Wrap-up** Review the computational thinking concept (Events) and the components (Canvas) learnt in this lesson.

Post-Lesson Activities	1. (Required) Complete the *Student Guide: Lesson 2* if not completed in class. 2. (Required) Ask students: "Have you played any similar games before?" (Pictionary) "How can you make this app work similarly?" 3. (Required) Ask students to think: "How can you make this drawing app more interesting to play with others?"

Lesson 3

Time	Activity
5 mins	**Review of Drawing App** 1. Review the drawing app by asking students how they can play this game with another person on a single device. 2. Ask students, "How can you play this game with another person at home?"
10 mins	**Demonstration of Communication with CloudDB** 1. Explain that whenever a user draws something on their device, they will also save the x,y coordinates for the start and end of the drag event to CloudDB. Other users will take that information and use it to draw the same line on their device. 2. Run the unplugged activity to role-play the sketcher, guesser, and CloudDB to demonstrate how the app will send drawing information from the sketcher to the guessers.
15 mins	**Adding CloudDB Component to the App** 1. Ask students to work using the Pair Programming model, following *Student Guide: Lesson 3*, to add CloudDB to their apps. (1) Students will add code to store drawing data in CloudDB.
5 mins	**Wrap-up** 1. Ask students what code is needed to allow the other apps users to see the drawing on their mobile device. (Students may say that the sketcher has stored the data, but the guesser must get the drawing data from CloudDB.) 2. Ask students if there is any app they have used before that shared data between users using CloudDB. 3. Explain that in Lesson 4, students will add the guesser code so the guessers see what is being drawn.

Post-Lesson Activities	1. (Required) Complete code if not finished. 2. Ask students to think about how they can code the app so the guesser gets the drawing information from CloudDB.

Lesson 4

Time	Activity
5 mins	**Review and Introduction to Lesson** 1. Ask students what parts they have coded so far in the app. (1) Drawing a picture on the screen. (2) Storing the drawing information in CloudDB. 2. Explain that in today's lesson, students will code the GuesserScreen, so the guesser will receive the drawing information from CloudDB.
25 mins	**Coding Activity** 1. Students code the GuesserScreen using the *Student Guide: Lesson 4* so the guesser receives the drawing information from CloudDB. 2. Students test the app to make sure that the guesser can see what the sketcher is drawing.
5 mins	**Wrap-up** 1. Ask students what else would make this game more interesting. (Students might suggest that the guesser can guess what is being drawn.) 2. Explain to students that they will implement functionality for guessing in Lesson 5. 3. Review computational thinking concepts covered in this lesson.
Post-Lesson Activities	(Required) Complete Student Guide if not completed in class.

Lesson 5

Time	Activity
5 mins	**Review and Introduction to Lesson** 1. Review the functions for the Sketcher and Guesser in the app. 2. Explain that students will start to code the guessing functionality in the app today.

25 mins	**Coding the Guessing Functionality** Following the *Student Guide: Lesson 5*, students add code to store the CurrentDrawing in CloudDB the app.
5 mins	**Wrap-up** 1. Ask students, “What is still missing from the app?” 2. Explain to students that they will add answer checking in the next lesson. 3. Review computational thinking concepts covered in this lesson.
Post-Lesson Activities	(Required) Complete code if not finished.

Lesson 6

Time	Activity
5 mins	**Introduction to Lesson** 1. Ask students what is missing from the existing Sketch and Guess app. 2. Teacher demonstrates the app with guess checking (SAG_MultiScreen_complete.aia) 3. Explain to students that they will implement guess checking in this lesson by adding code to provide feedback to users when they guess.
20 mins	**Coding New Features** 1. Explain to students that they will use the Spinner component. 2. Following the *Student Guide: Lesson 6*, students add guess checking to their apps. 3. Student groups test the app for the new functionality. 4. Any student pairs who complete Lesson 6 coding may attempt the *Student Guide: Challenge* where they can add colour buttons and a slider to change the line width in the app.
10 mins	**Wrap-up** 1. Review the use of CloudDB in this unit. 2. Review computational thinking concepts covered in this unit. 3. Review computational thinking practices used in this unit.
Post-Lesson Activities	1. (Required) Answer online multiple-choice questions. 2. (Required) Complete survey of learning attitudes. 3. (Optional) Add new functionality to app (countdown timers, scoring).

7. Instructional Details

The teaching and learning materials of the six lessons are presented in the appendices below. Please note the inclusion of the template and checkpoint aia files. These MIT App Inventor project files are included as backup resources for teachers. Each checkpoint aia files includes the project and code blocks that are expected to be completed during that lesson. If necessary, teachers can use these as reference or even a start project file for students who miss a lesson due to illness.

Lesson 1

1. Teacher’s Guide: Lesson 1
2. Student Guide: Lesson 1
3. SAG_MultiScreen_complete.aia
4. SAG_MultiScreen_complete.apk
5. SAG_MultiScreen_template.aia

Lesson 2

1. Teacher’s Guide: Lesson 2
2. Student Guide: Lesson 2

Lesson 3

1. Teacher’s Guide: Lesson 3
2. Student Guide: Lesson 3
3. SAG_MultiScreen_checkpoint2.aia

Lesson 4

1. Teacher’s Guide: Lesson 4
2. Student Guide: Lesson 4
3. SAG_MultiScreen_checkpoint3.aia

Lesson 5

1. Teacher’s Guide: Lesson 5
2. Student Guide: Lesson 5
3. SAG_MultiScreen_checkpoint4.aia

Lesson 6

1. Teacher’s Guide: Lesson 6
2. Student Guide: Lesson 6
3. SAG_MultiScreen_checkpoint5.aia
4. SAG_MultiScreen_complete.aia
5. Student Guide: Challenge
6. SAG_MultiScreen_challenge.aia

8. Assessment

The following table maps the assessment modes and corresponding Unit Intended Learning Outcomes (UILOs) in Section 2.

Assessment Modes	UILO
Self-assessment: (1) Online multiple-choice questions (2) Survey of learning attitudes	$UILO_{2,3}$

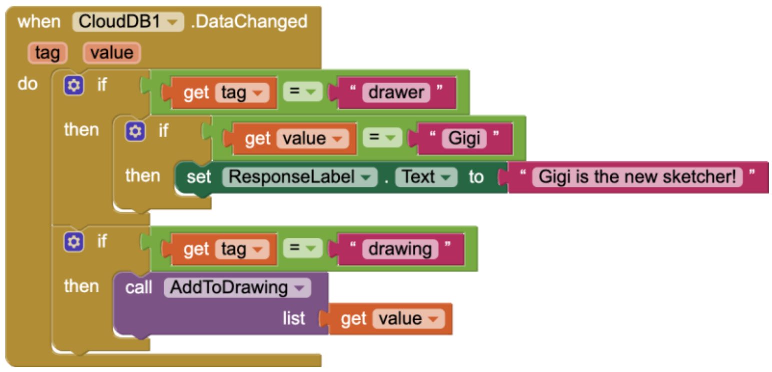
Self-assessment

Self-assessment allows students to reflect on their learning performance. There are two sorts of self-assessment questions below:

Online Multiple-choice Questions

Online multiple-choice questions are adopted to assess students' understanding of the key concepts of the activity, as shown below.

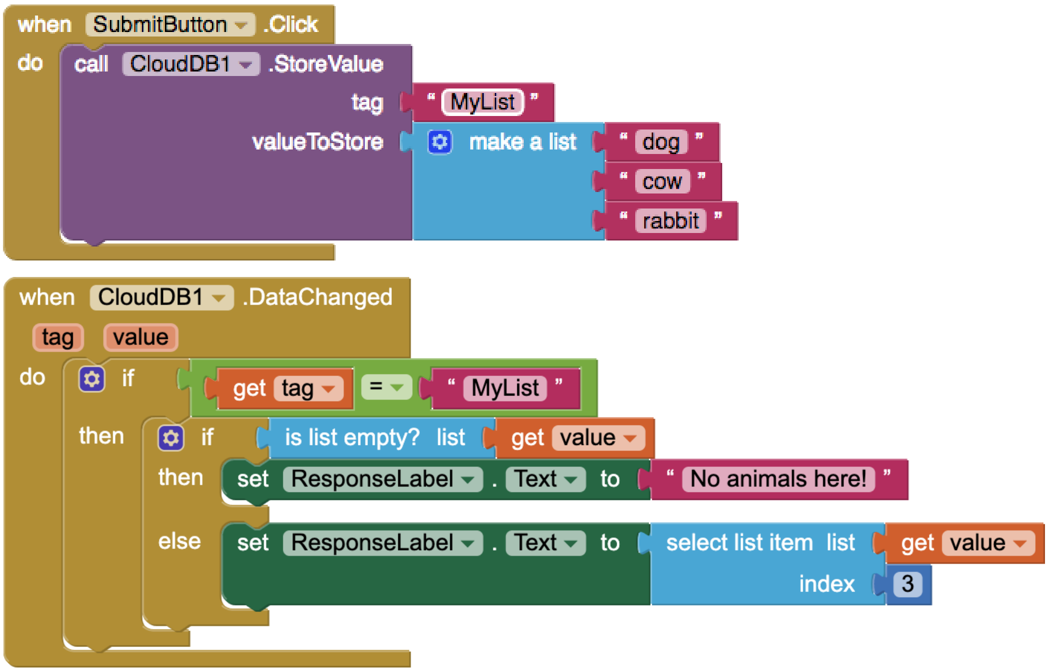
1. If the following code is used and the tag is "drawer" and value is "Henry", what will happen in the app?



A. "Gigi is the new drawer!" will be displayed.
B. "Henry is the new drawer!" will be displayed.
C. The AddToDrawing procedure is called.
D. Nothing.

(Answer: D)

2. The following code is used in an app. The user presses the Submit button. What should happen?



A. “dog” is displayed.

B. “cow” is displayed.

C. “rabbit” is displayed.

D. “No animals here” is displayed.

E. Nothing.

(Answer: C)

Survey of Learning Attitudes

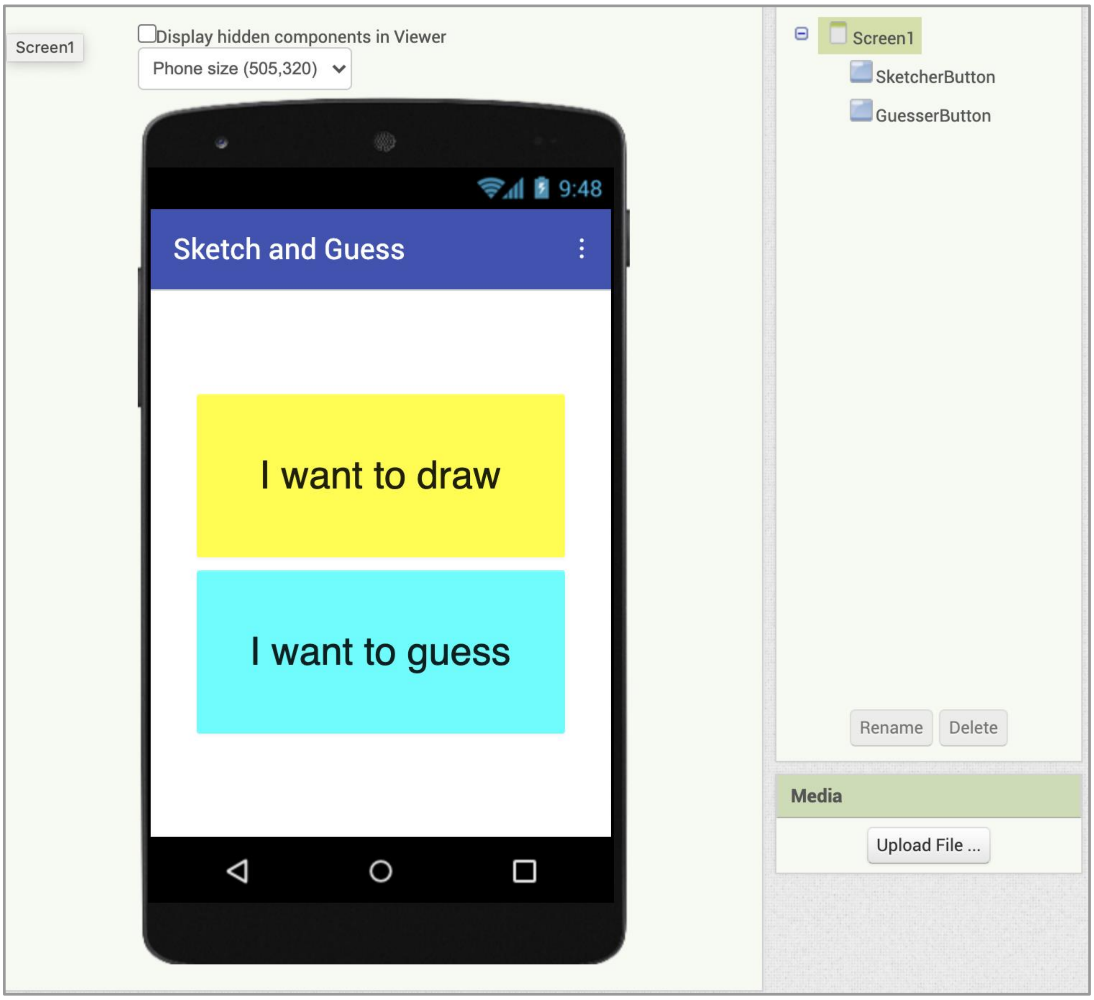
To evaluate students' attitudes, perceptions and understandings of coding, they are required to complete the following questionnaire by putting a "✓" in the appropriate boxes.

After completion of this unit, I think…	**Disagree**	**Somewhat disagree**	**Neutral**	**Somewhat agree**	**Agree**
I feel inspired when learning more about programming.					
After learning programming, I have a deeper understanding of how digital technology works.					
Programming helps me relate to people in my community.					
The learning activities of this unit are interesting and I like them.					
I am excited to share this app with friends and family.					

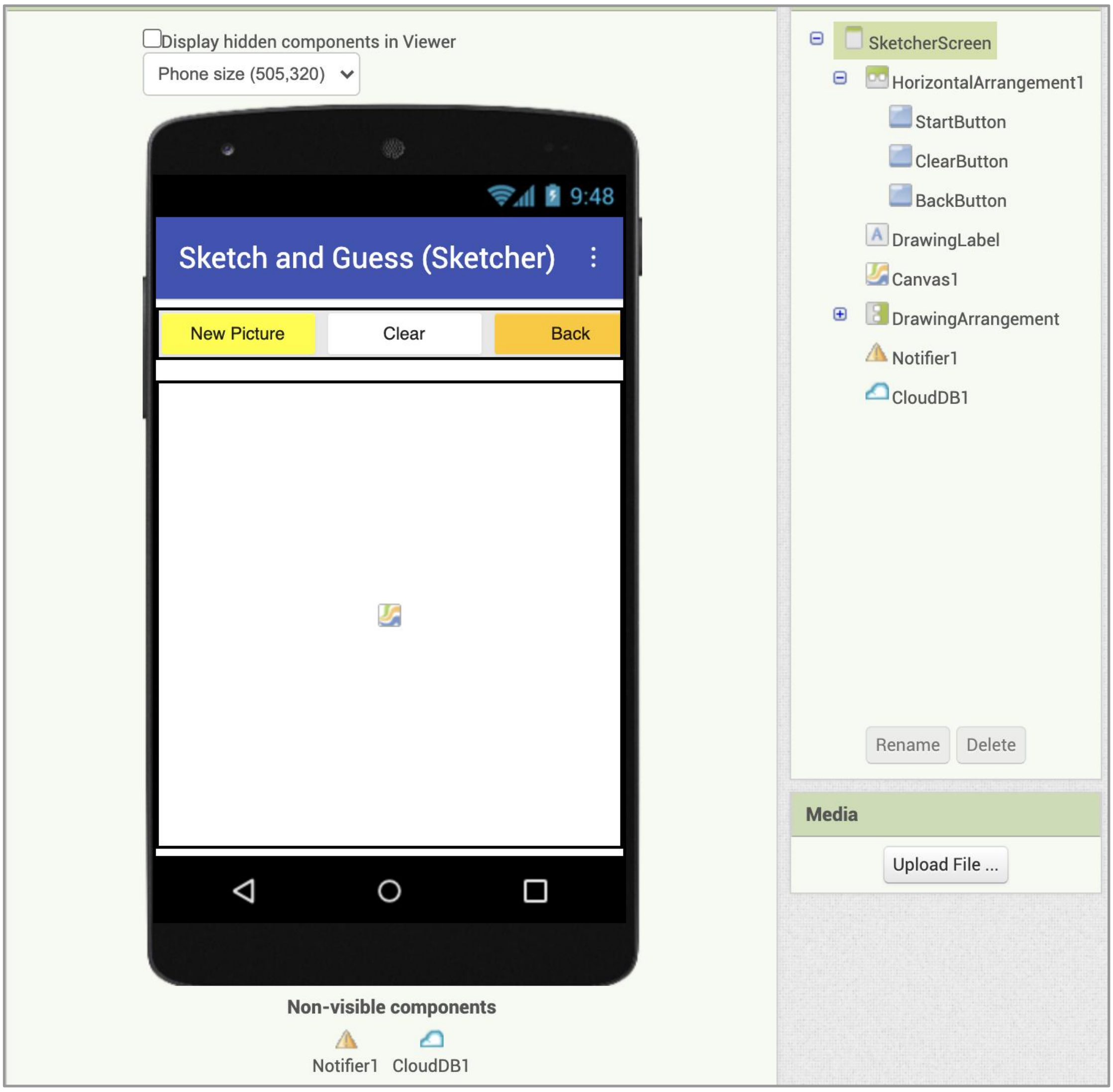
9. Screen Design and Code

Designer

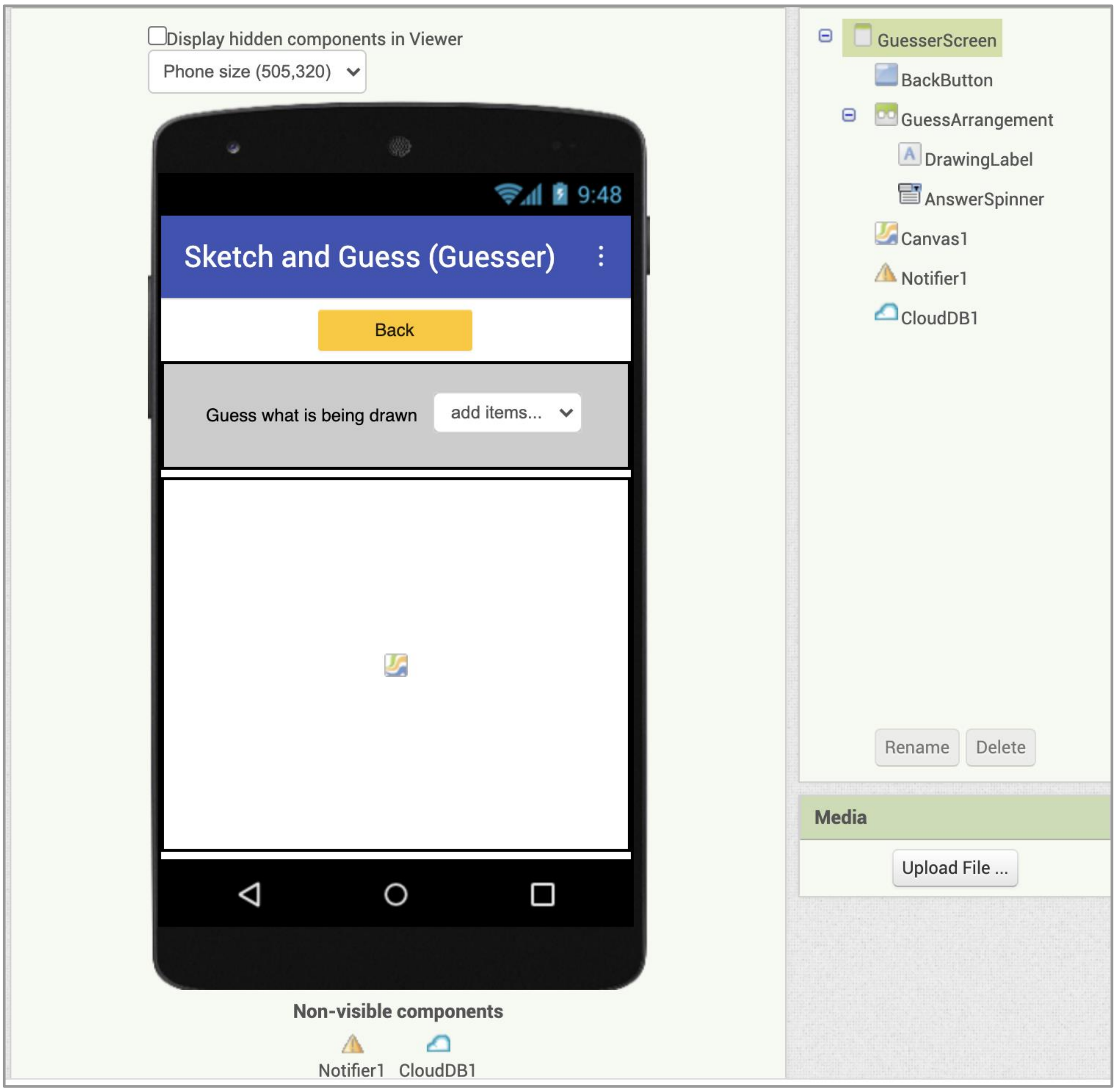
Screen1



SketcherScreen



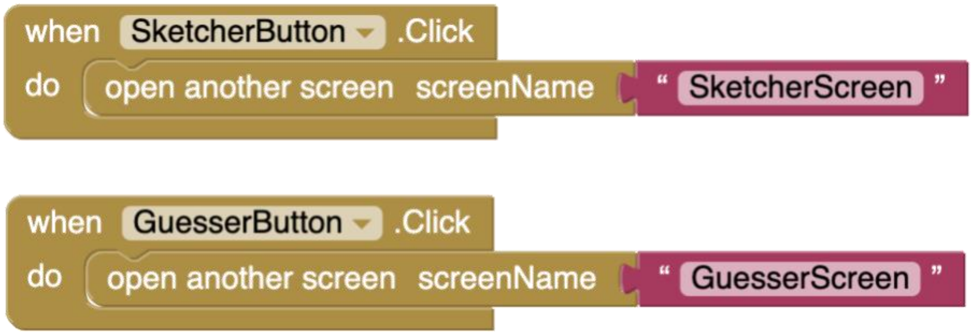
GuesserScreen



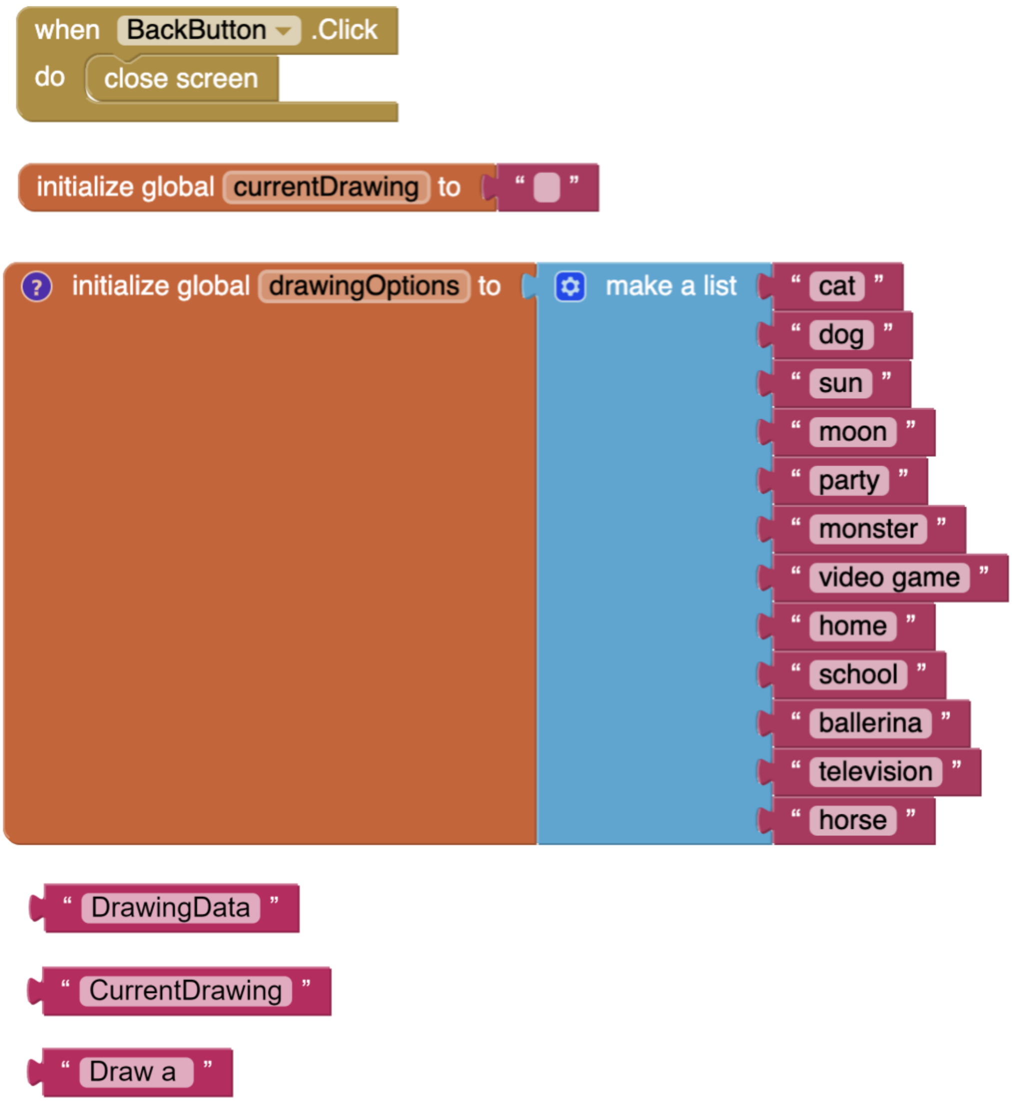
Blocks

Template:

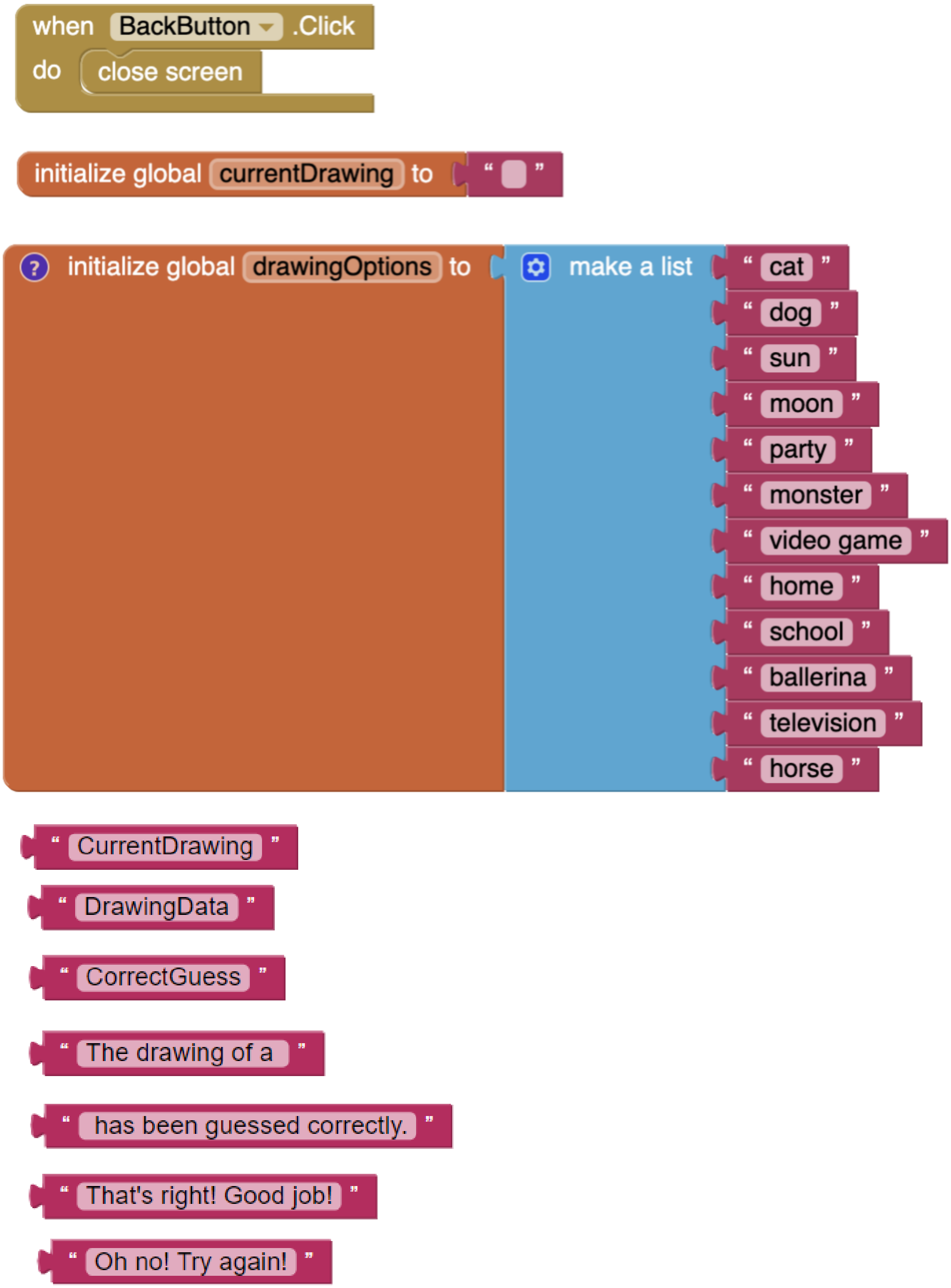
Screen1



SketcherScreen

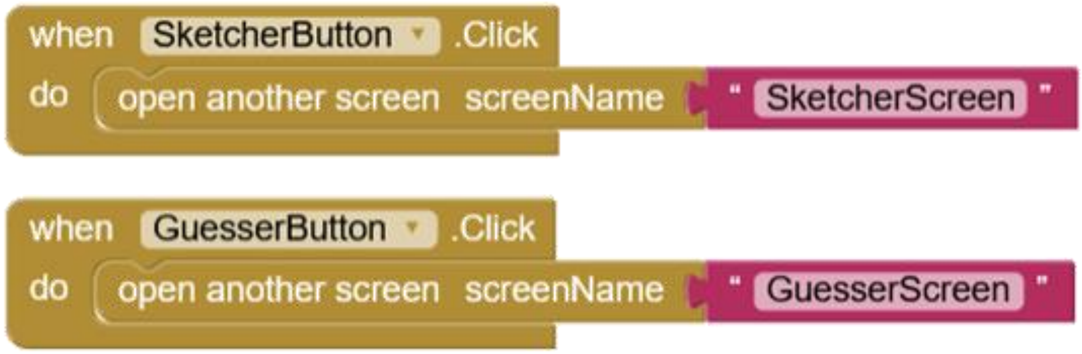


GuesserScreen

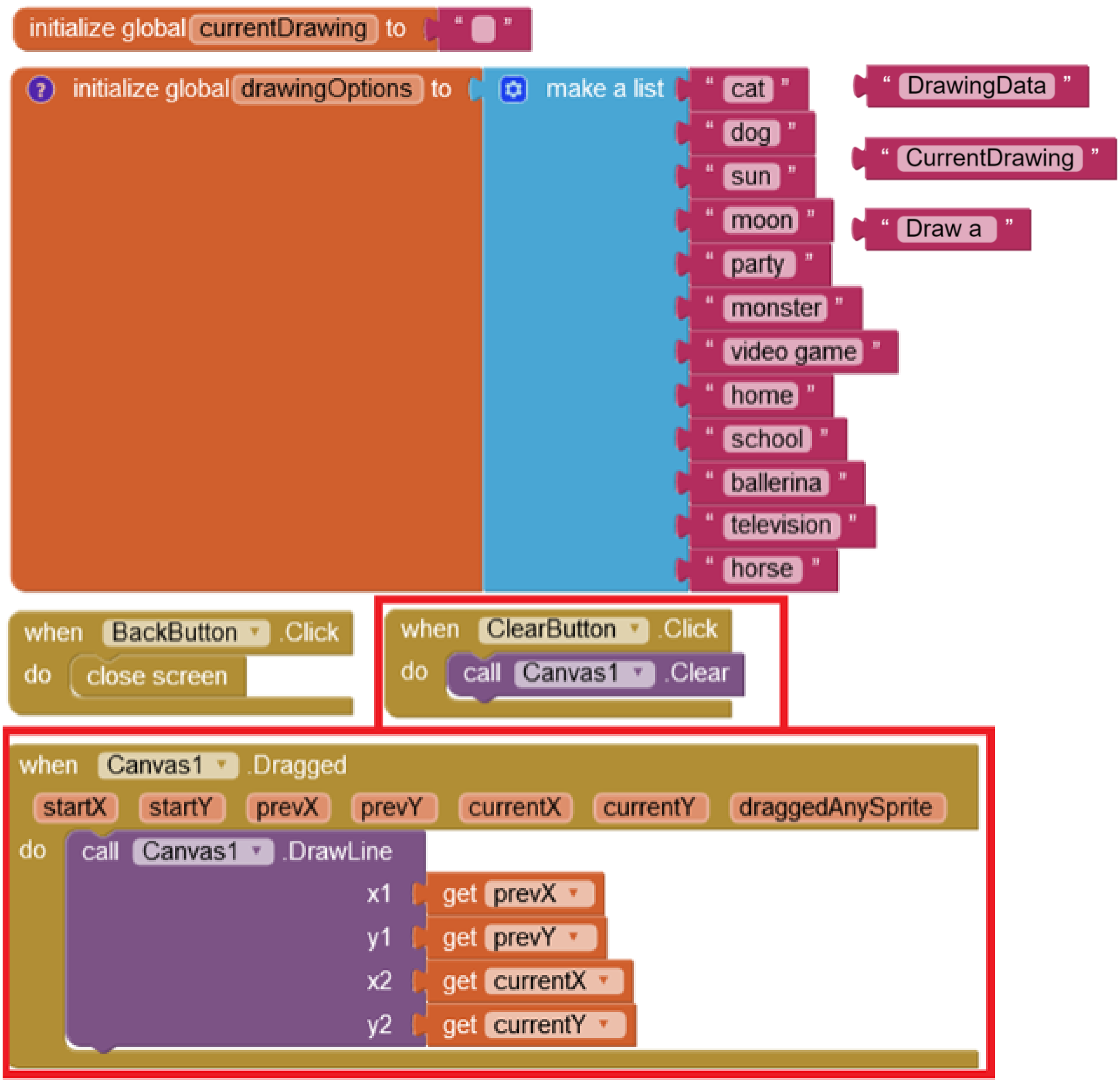


Lesson 2 (SAG_MultiScreen_checkpoint2): new blocks are highlighted in red

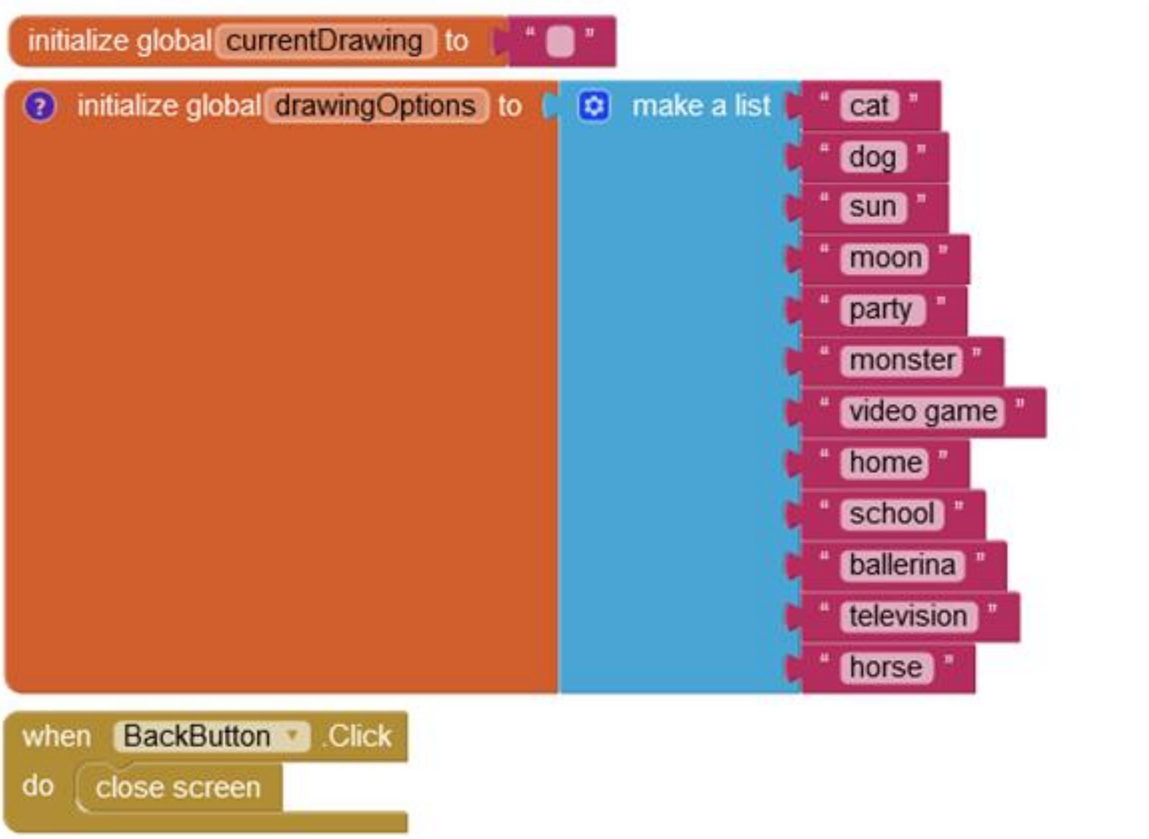
Screen1

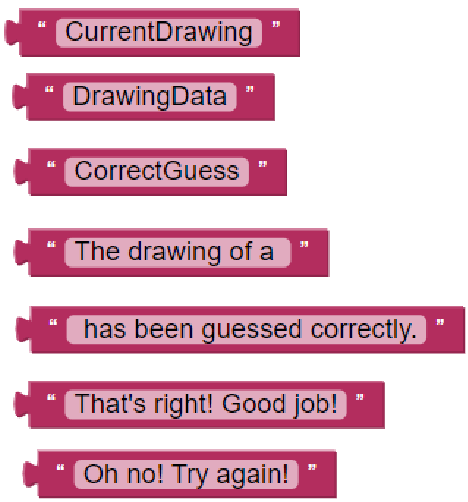


SketcherScreen



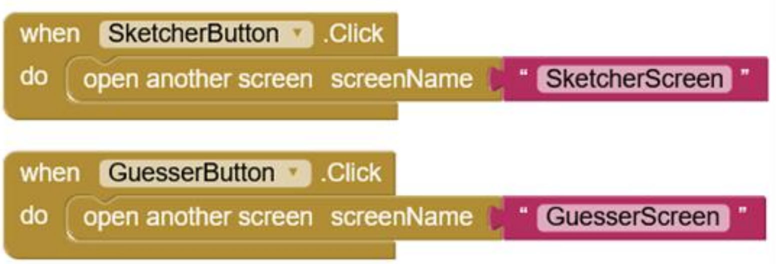
GuesserScreen



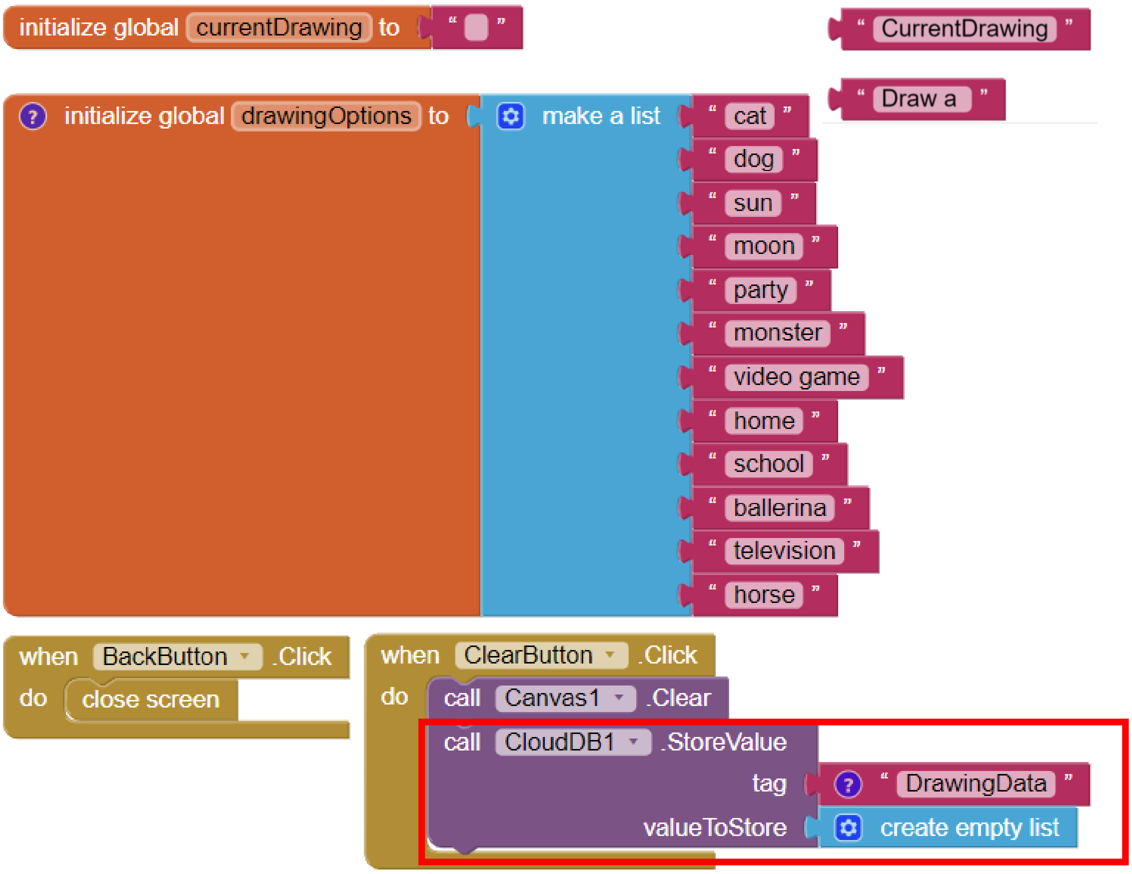


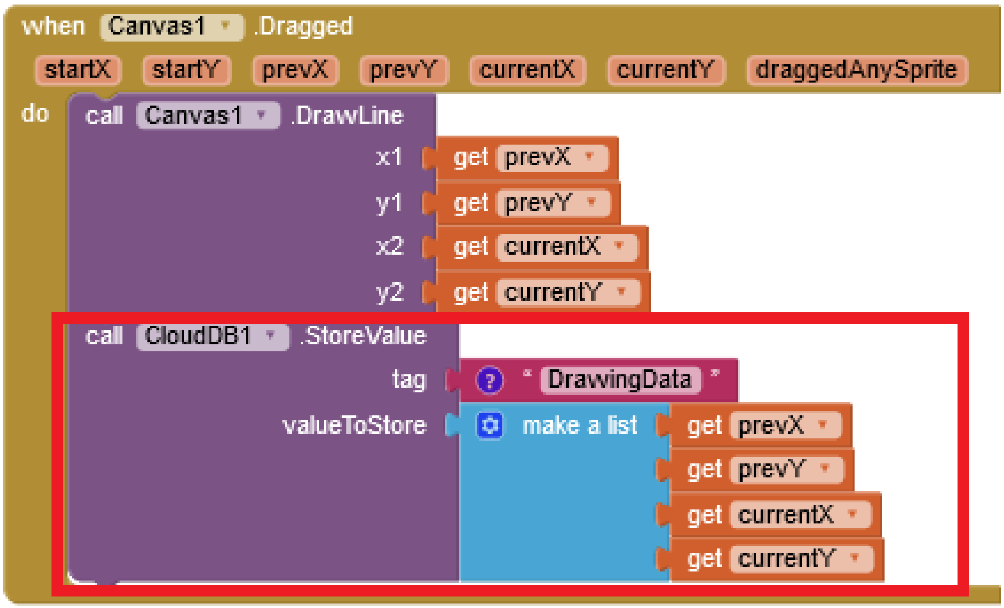
Lesson 3 (SAG_MultiScreen_checkpoint3)

Screen1

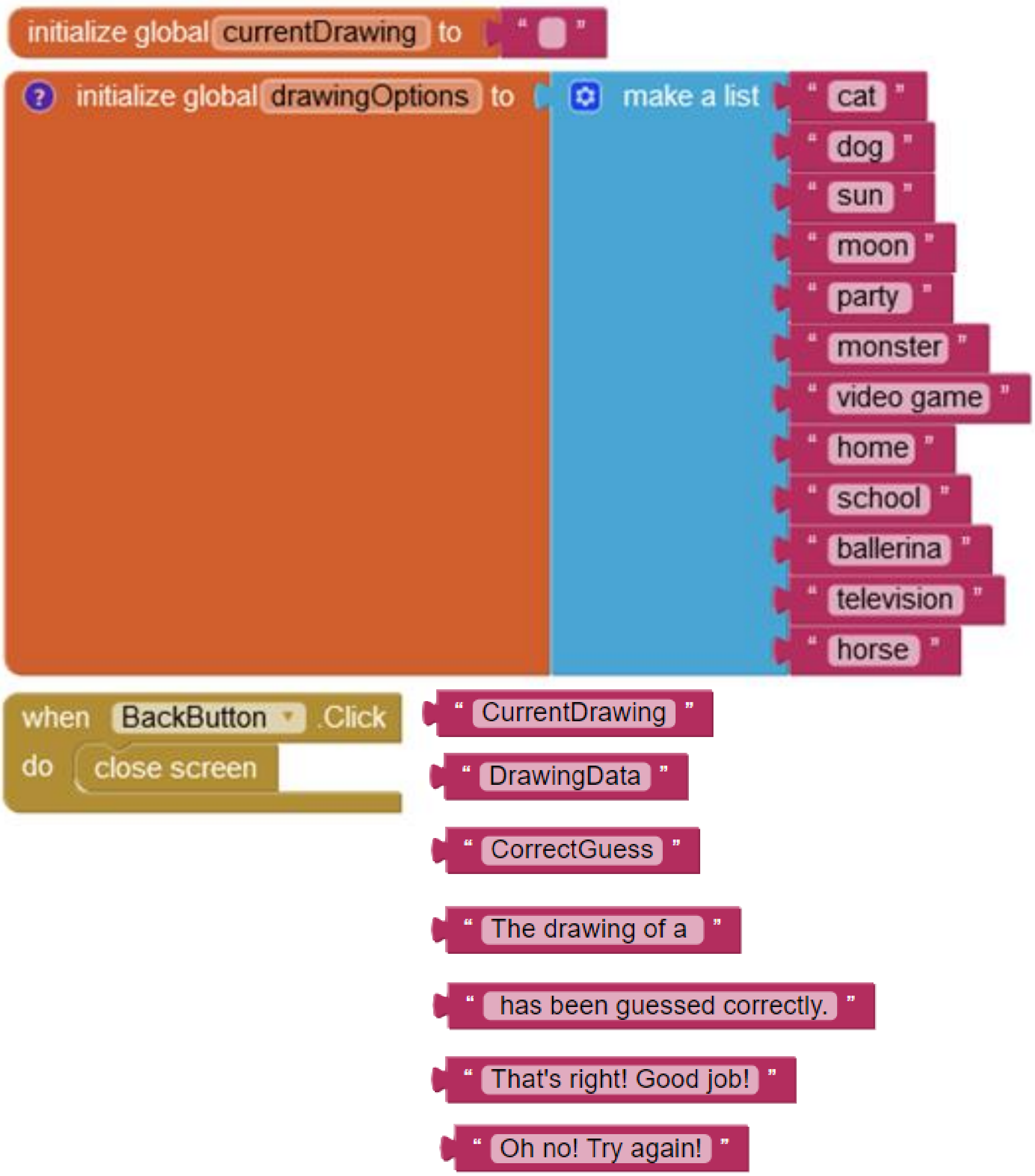


SketcherScreen



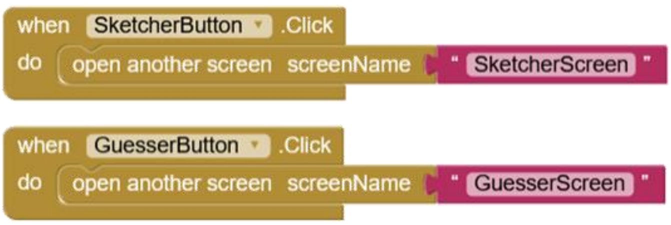


GuesserScreen

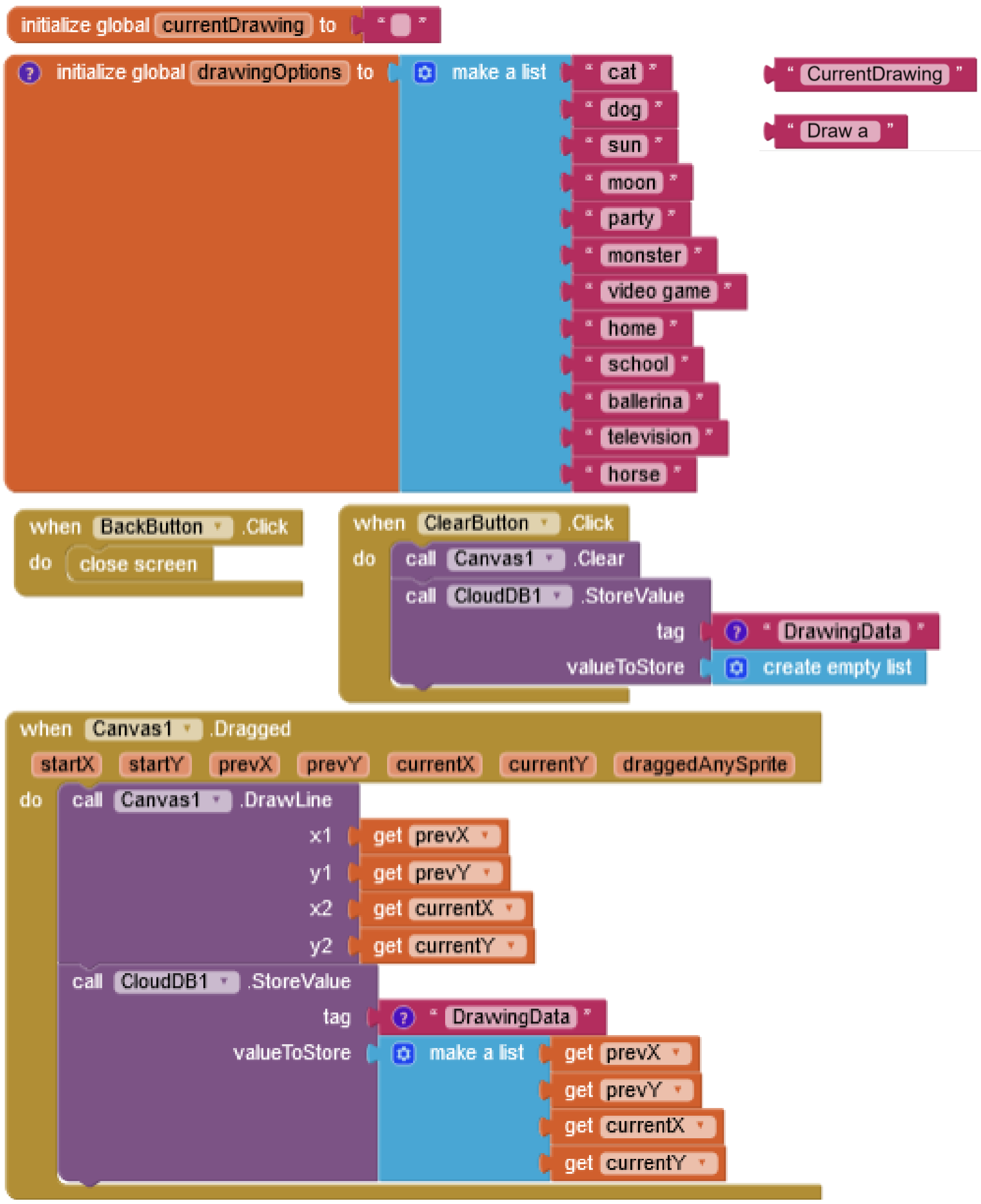


Lesson 4 (SAG_MultiScreen_checkpoint4)

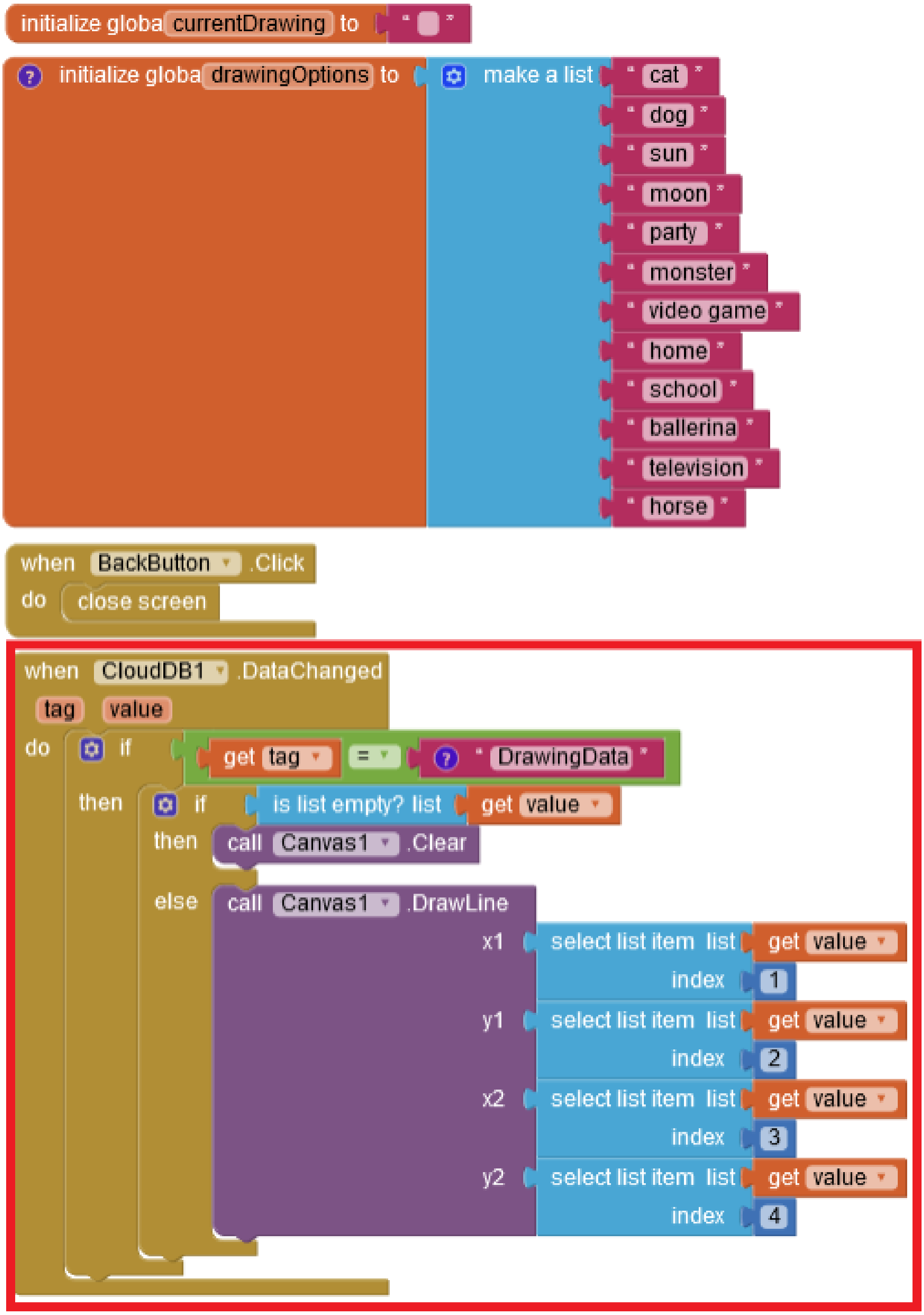
Screen1

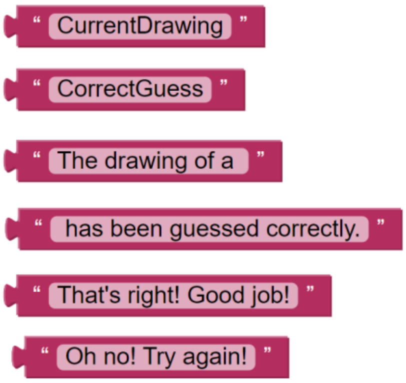


SketcherScreen



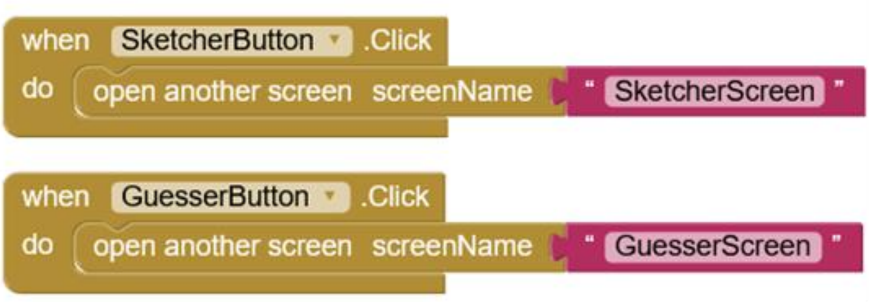
GuesserScreen



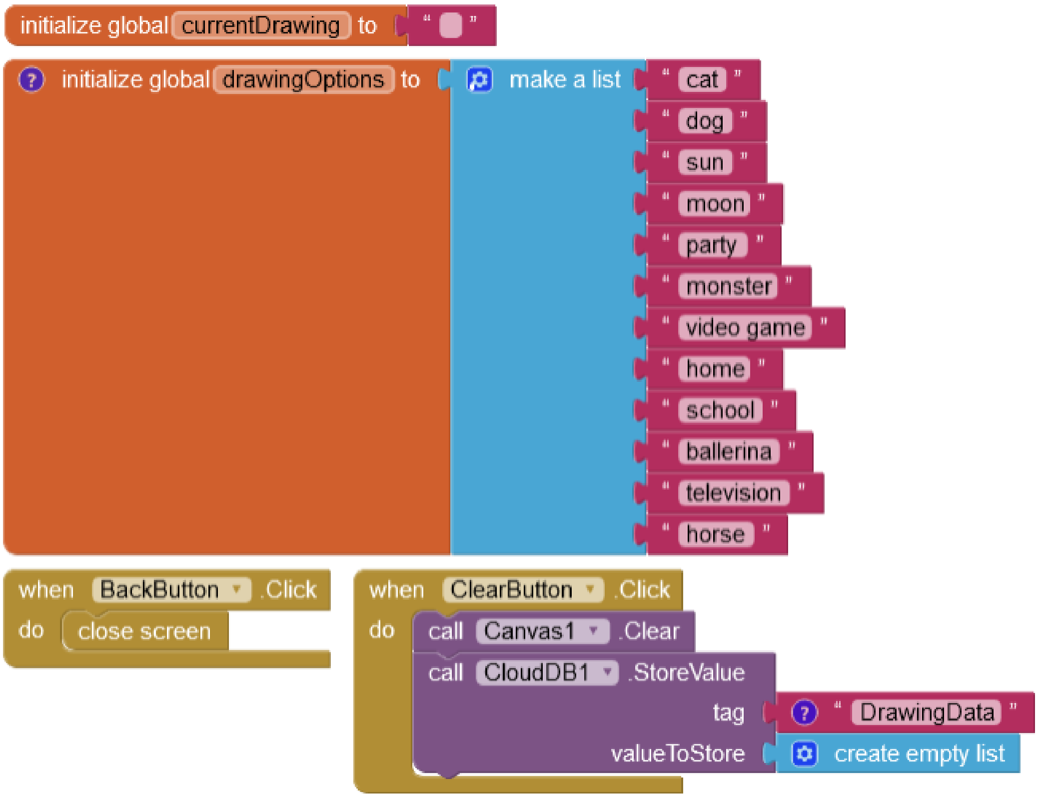

" CurrentDrawing "
" CorrectGuess "
" The drawing of a "
" has been guessed correctly. "
" That's right! Good job! "
" Oh no! Try again! "

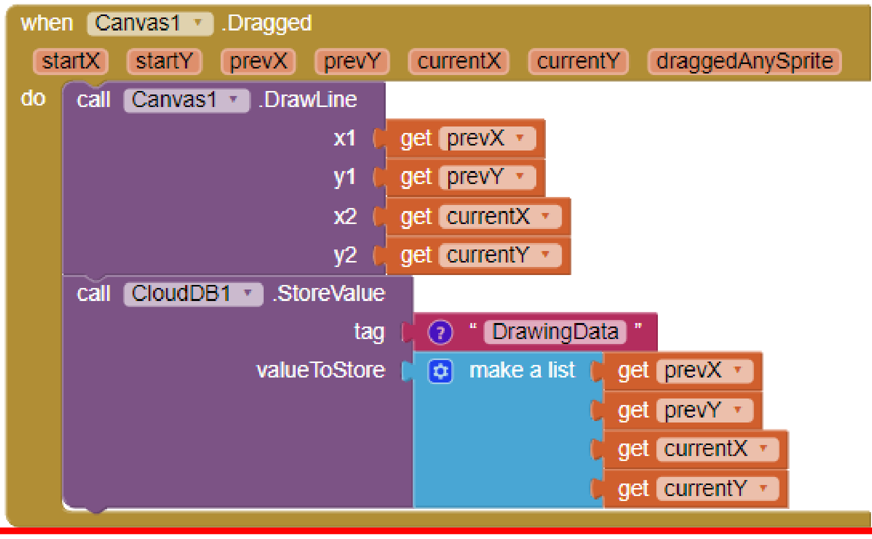
Lesson 5 (SAG_MultiScreen_checkpoint5)

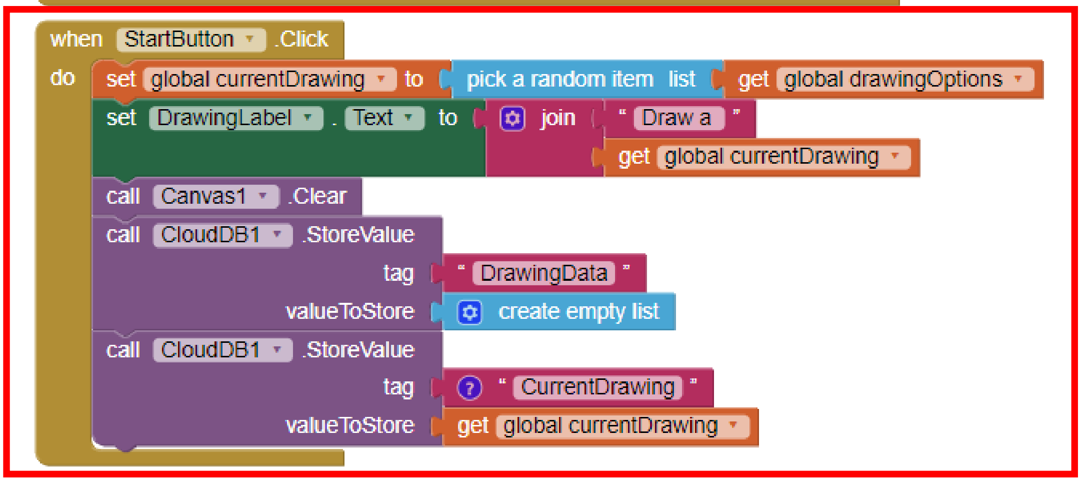
Screen1



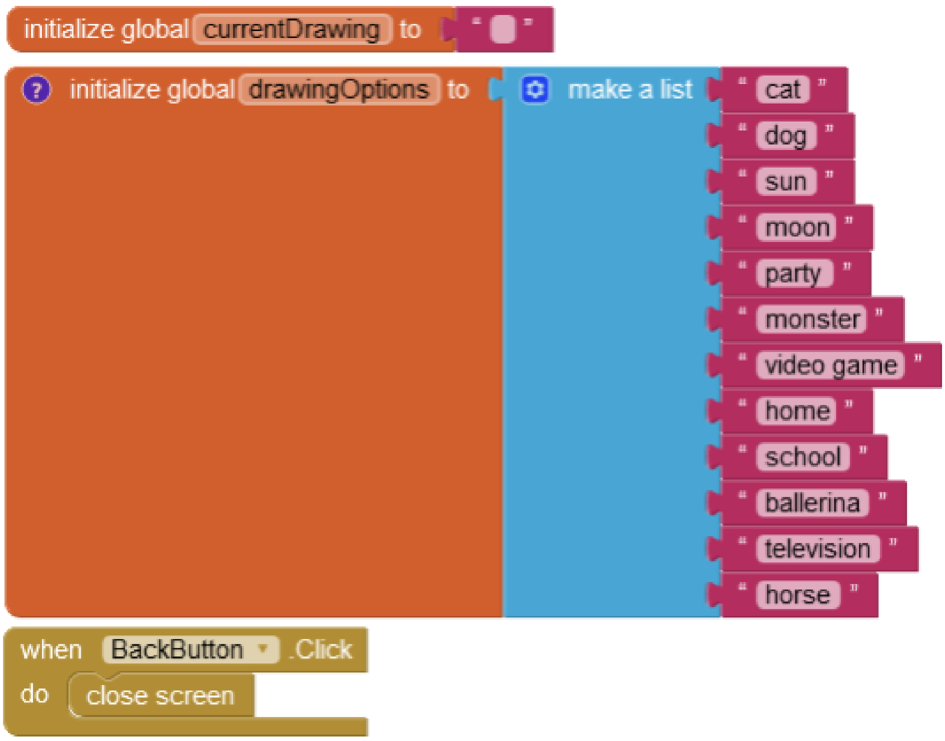
SketcherScreen

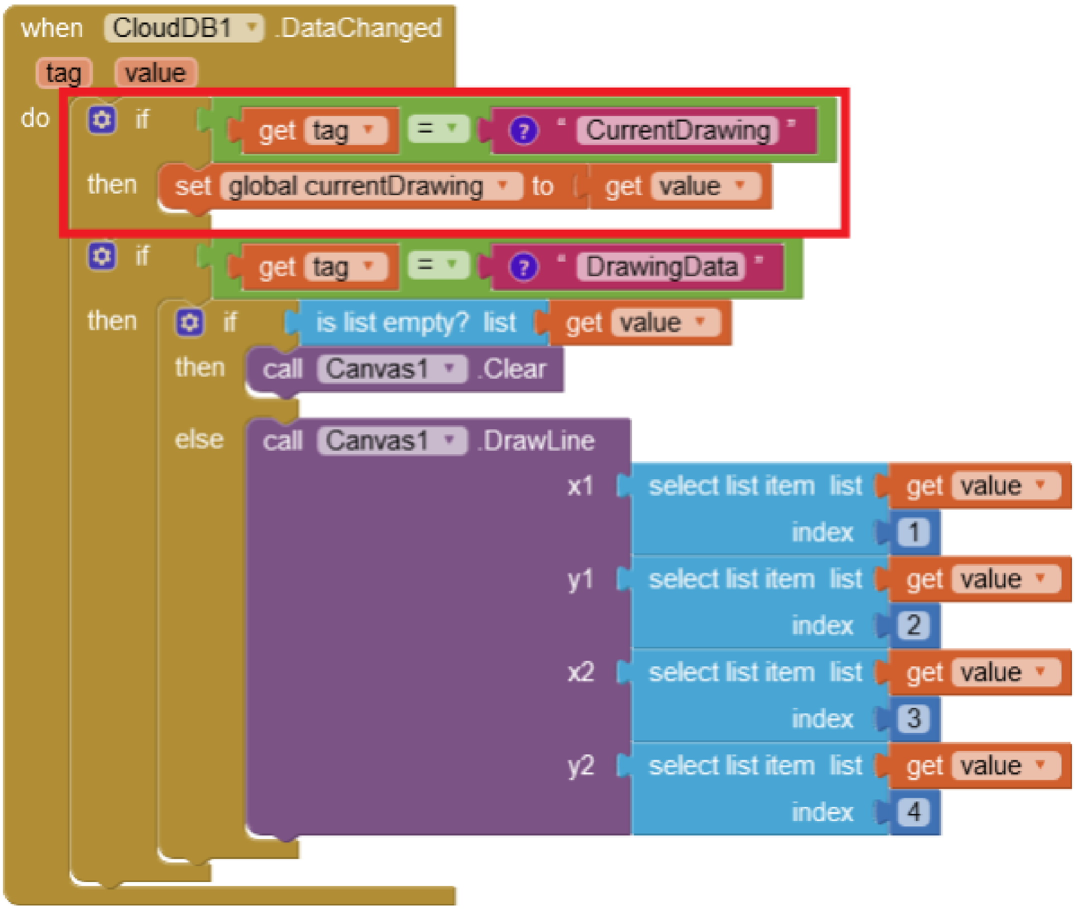


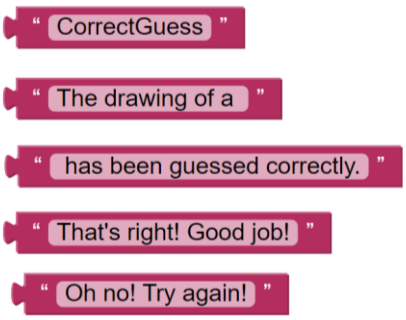




GuesserScreen

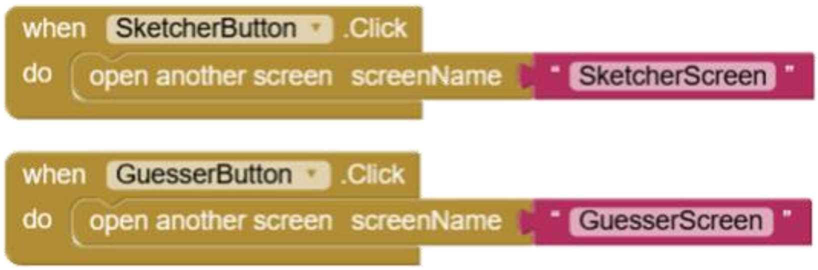



when CloudDB1 .DataChanged
tag value
do if get tag = " CurrentDrawing "
then set global currentDrawing to get value
if get tag = " DrawingData "
then if is list empty? list get value
then call Canvas1 .Clear
else call Canvas1 .DrawLine
x1 select list item list get value
index 1
y1 select list item list get value
index 2
x2 select list item list get value
index 3
y2 select list item list get value
index 4

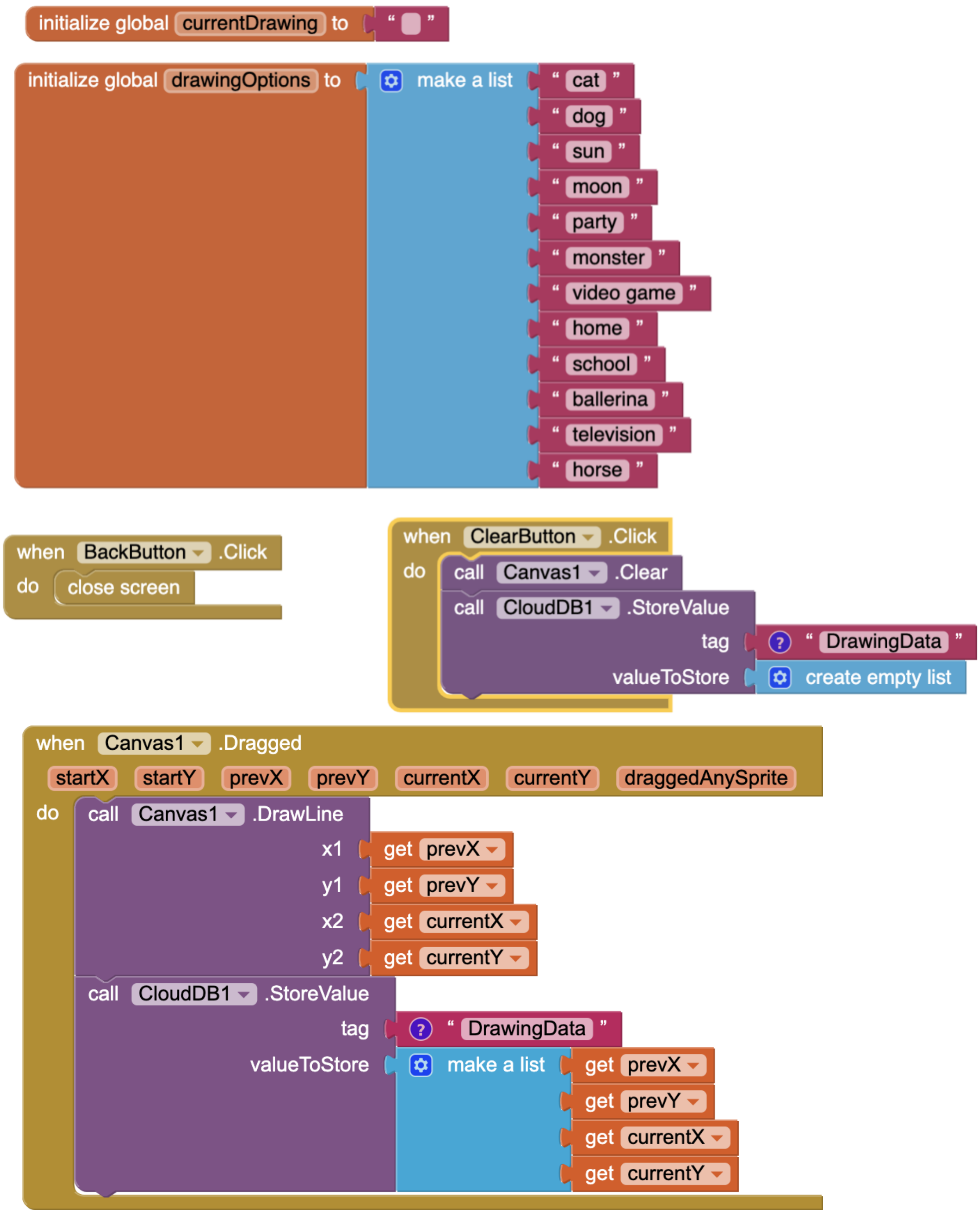

" CorrectGuess "
" The drawing of a "
" has been guessed correctly. "
" That's right! Good job! "
" Oh no! Try again! "

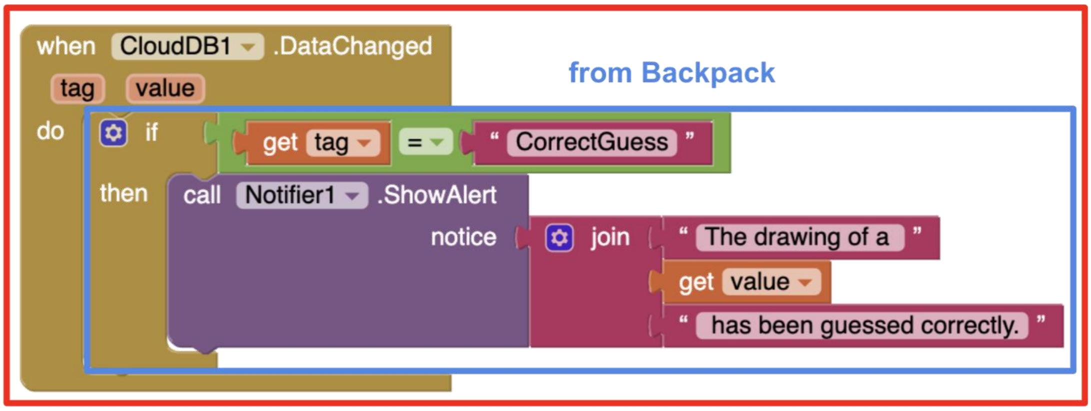
Lesson 6 (SAG_MultiScreen_complete)

Screen1

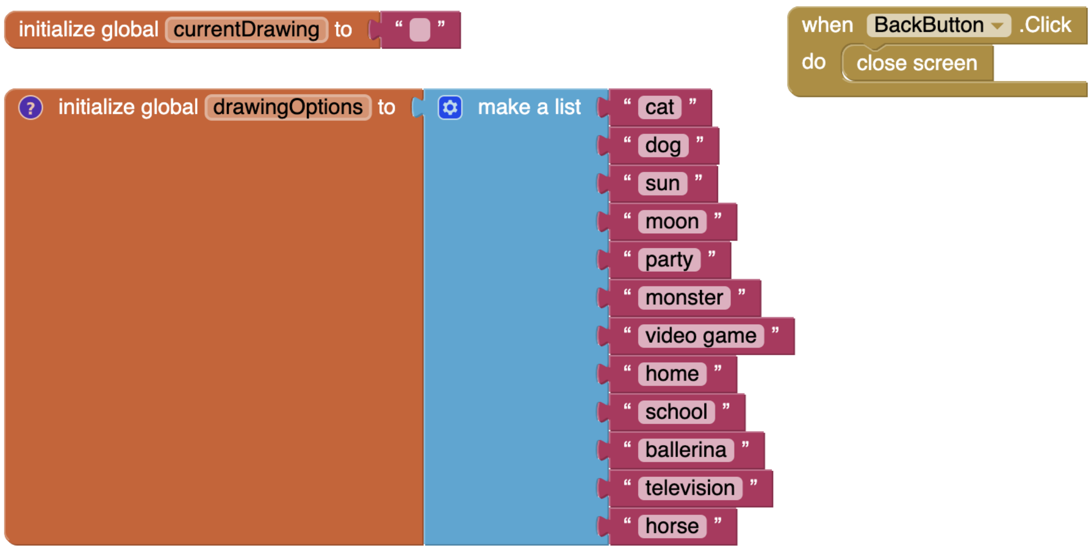


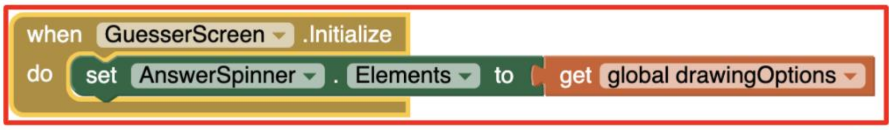
SketcherScreen

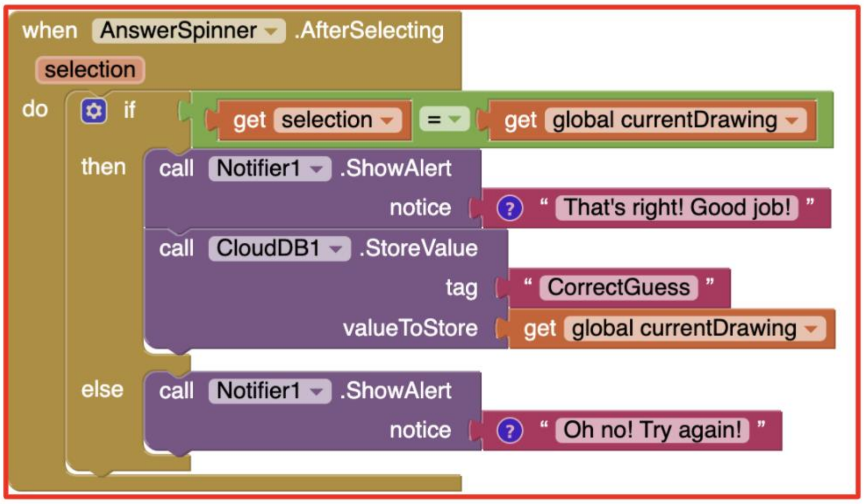


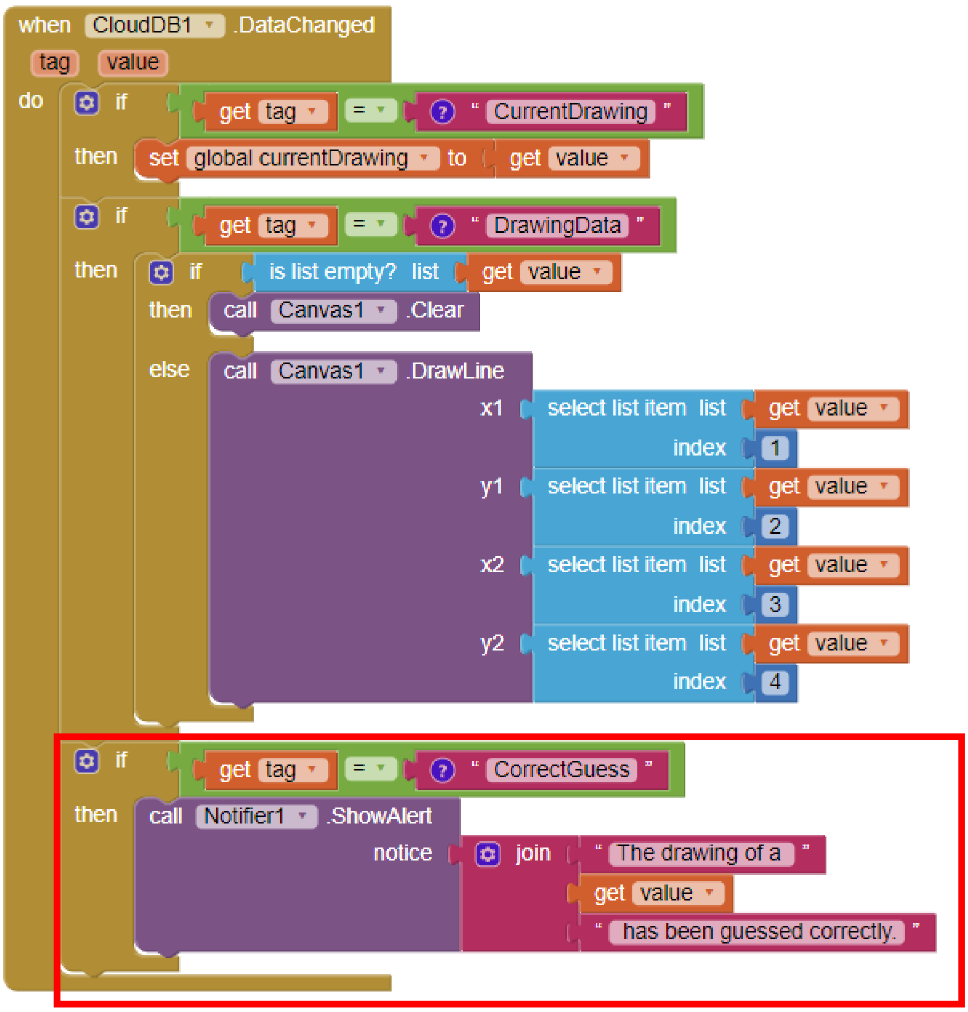


GuesserScreen







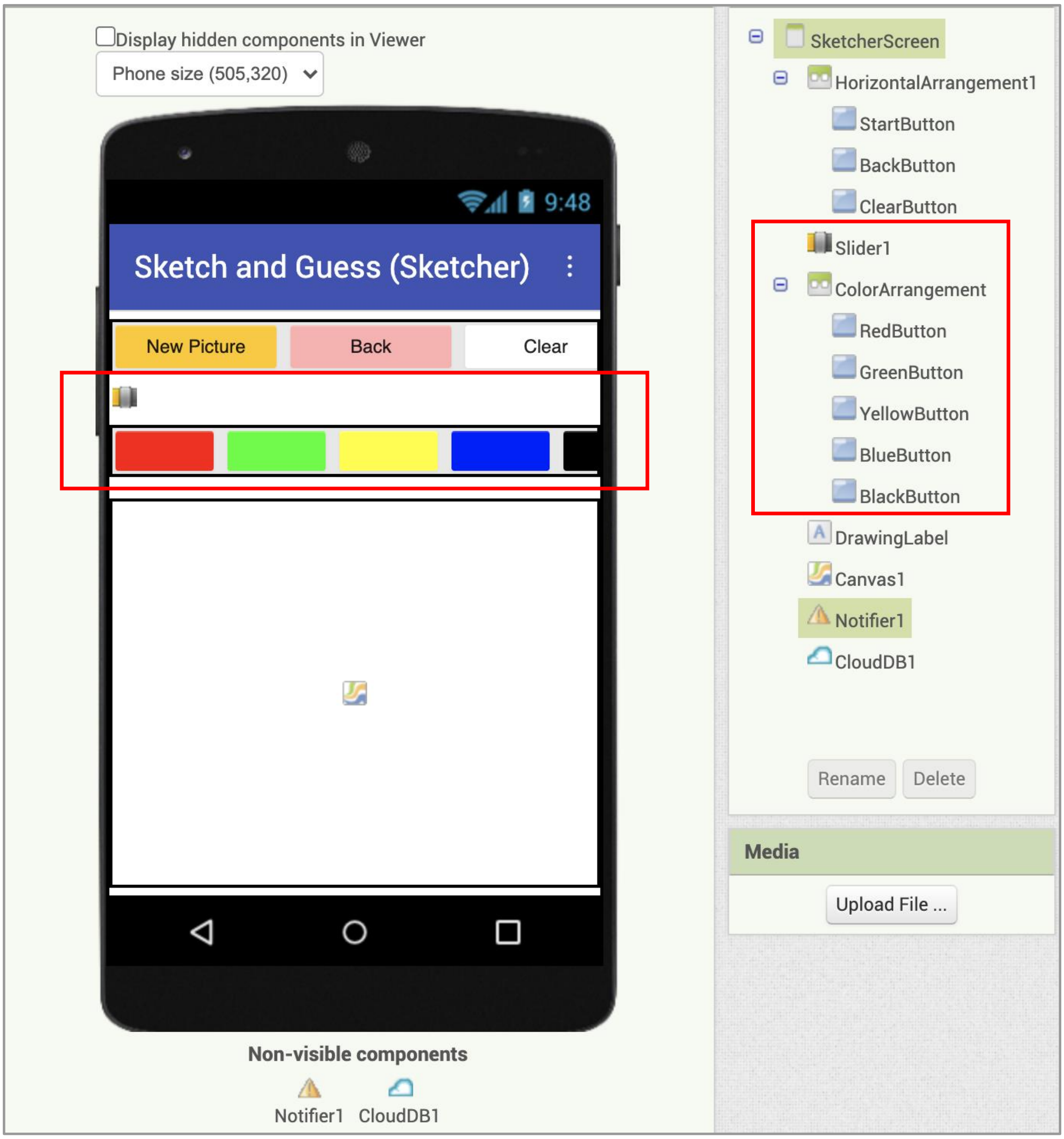


Save to Backpack

Challenge (SAG_MultiScreen_challenge)

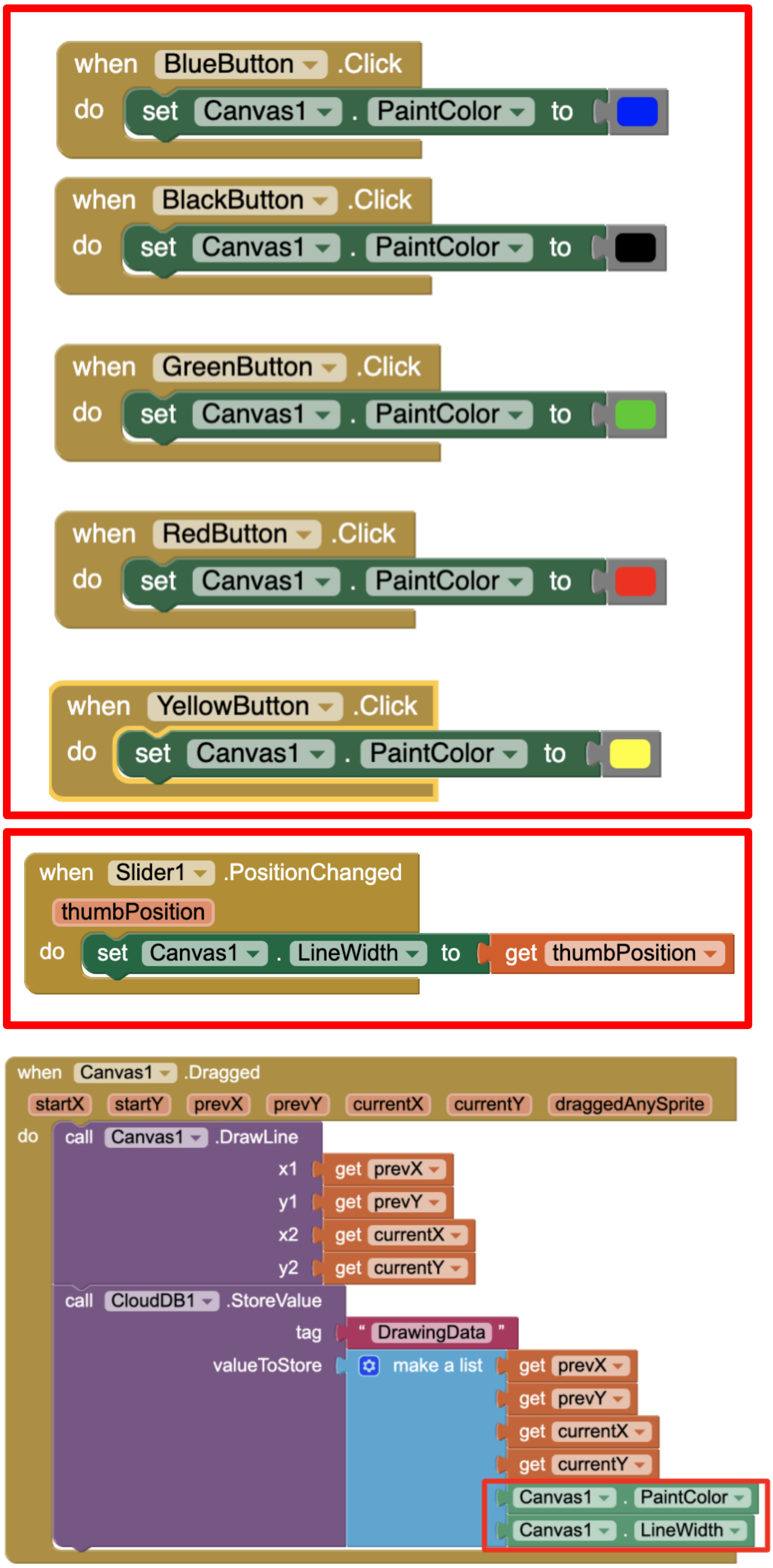
Designer:

SketcherScreen

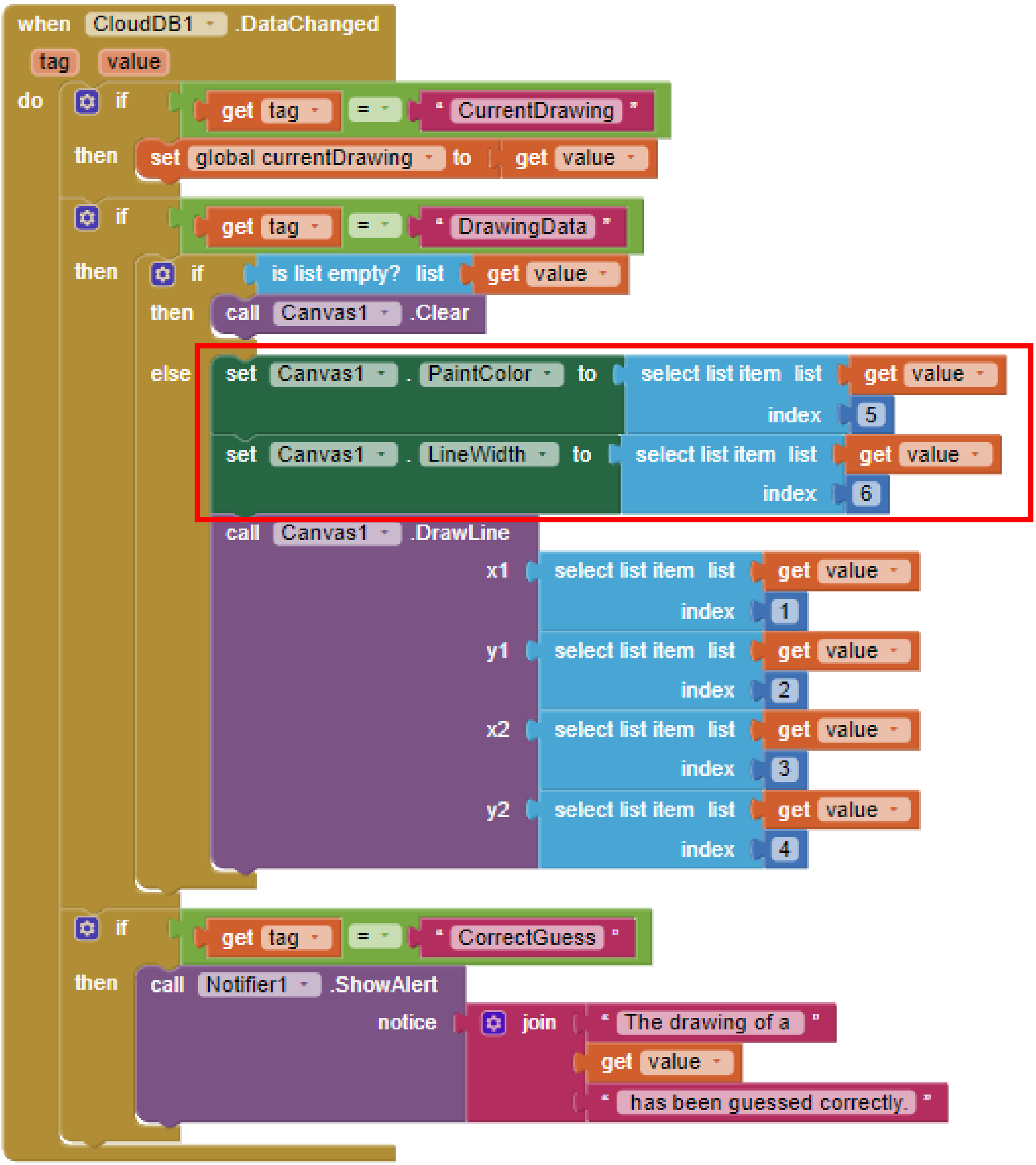


Blocks

SketcherScreen (added blocks are shown, red highlights show added blocks within existing events)



GuesserScreen (red highlights show added blocks within existing events)



Appendix 1

L3U8.6-8.8 Teacher's Guide: Lesson 1

Teaching Objectives

1. Raise students' interest in building a drawing app.
2. Focus: Teach students how use the computational practice of algorithmic thinking and abstracting and modularizing to decompose an app into separate tasks.
3. Encourage students to work collaboratively to create and test an app.

Lesson Outline

Introduction to Drawing with Canvas Component (10 minutes)

Demonstrate the completed app (SAG_MultiScreen_complete.aia). Let students play the drawing app to see how it works.

Review and Introduction to Lesson (5 minutes)

1. Review the flow of the app.
2. Students complete Page 1 of *Student Guide: Lesson 1*. Fill out a sequencing diagram to decompose. Here is the diagram with completed answers.

A. The Sketcher draws the item.

B. A randomly generated item is displayed on the Sketcher's screen.

C. The sketch appears on the Guessers' screen.

D. Users press"I want to draw" and "I want to guess" buttons respectively based on their verbal agreement.

E. The Guesser selects item from dropdown list to guess what the Sketcher has drawn

F. The app checks if the guess is correct and displays appropriate message.

D	B	A	C	E	F

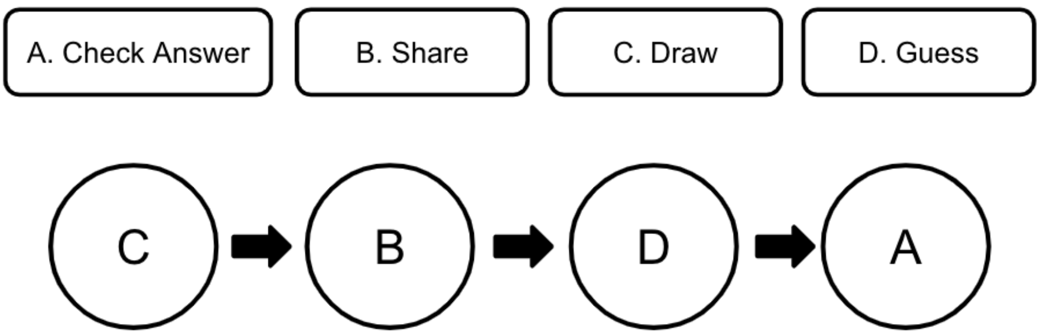
Decomposition of Parts of App (20 minutes)

1. Ask students if they think this is a complex app to build.
2. Explain that it helps to break down the parts of a big app into smaller parts, and then build each part, one at a time. Students should always test and make sure the code works as expected before moving forward to the next step in development.
3. Ask students to complete the *Student Guide: Lesson 1*.

   (1) List out the tasks and abilities of each player in the game.

Sketcher		**Guesser**
• Can draw • Can clear canvas • See the drawing	• Sees displayed item to draw • Can draw • Cannot draw • Can clear canvas • Cannot clear canvas • See the drawing • Sees displayed list of items to guess	• Sees displayed item to draw • Cannot draw • Cannot clear canvas • See the drawing • Sees displayed list of items to guess

Here is a completed diagram from the Student Guide, page 3.



   (2) Test the opening and closing of screens in the project template.

Post-Lesson Activities

(Required) Complete the *Student Guide: Lesson 1* if not completed in class.

Appendix 2

L3U8.6-8.8 Teacher's Guide: Lesson 2

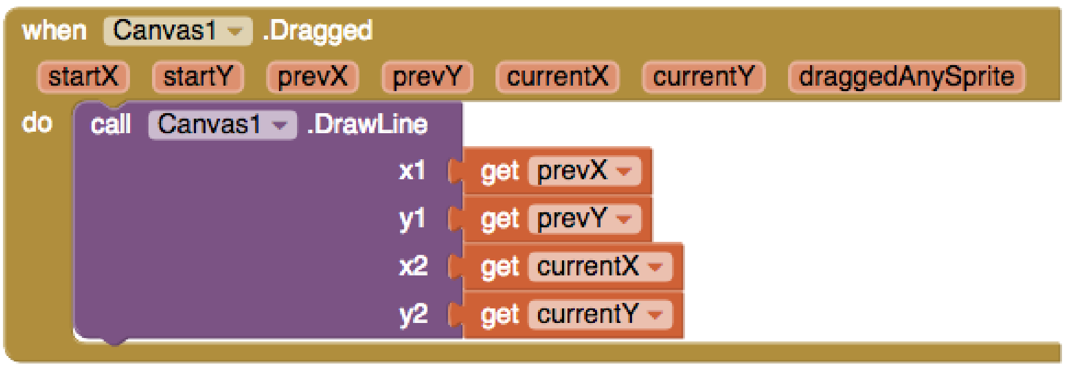
Teaching Objectives

1. Focus: Guide students how to use the computational thinking practice of algorithmic thinking to add drawing capability to a mobile app.
2. Encourage students to work collaboratively to build and test a multiplayer game.

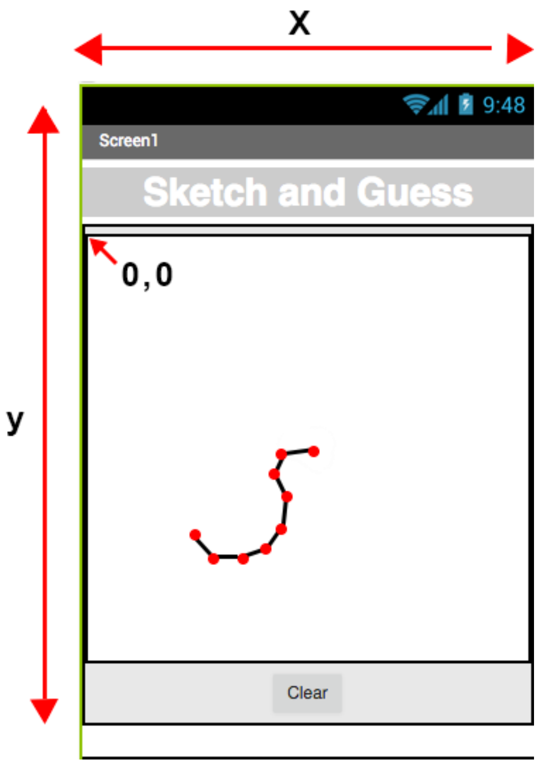
Lesson Outline

Introduction to Drawing with Canvas Component (10 minutes)

1. Explain to students that they will code the drawing feature of the app in this lesson. Ask students to fill out Page 1 of the *Student Guide: Lesson 2*.
2. Show students how a line is drawn on the canvas based on a user dragging their finger on the screen.
   (1) Remind students of the canvas coordinate system.
   (2) Show students the code blocks for Canvas1.DrawLine. Using the Canvas.DrawLine event, which uses the start and end drag points (prevX, prevY to currentX, currentY) to draw a line, and connecting those lines together, a fluid line will appear on the canvas.



(3) Explain that the input parameters represent the x and y coordinates on the canvas where the user starts and ends the dragging of their finger.



(4) Ask students to fill out pages 2, 3 of the *Student Guide: Lesson 2*.

Coding of Simple Drawing App (20 minutes)

1. Ask students to build a drawing app with the help of the *Student Guide: Lesson 1* and the template aia file.
2. Explain to students that they will work using the Pair Programming model on this unit. Remind them how the Pair Programming model works, and ask them if they can remember the table of rules:

DO	**DON'T**
• Be respectful; • Talk to one another about the work; • Explain what you are doing; • Think ahead and make suggestions; • Switch roles often.	• Be a bossy navigator; • Grab the driver's mouse/keyboard.

3. Students should test and debug the app to see that they are able to draw on the canvas.

Wrap-up (5 minutes)

Review the computational thinking concept (Events) and the component (Canvas) learnt in this lesson.

Post-Lesson Activities

1. (Required) Complete the *Student Guide: Lesson 2* if not completed in class.
2. (Required) Ask students: "Have you played any similar games before?" (Pictionary) "How can you make this app work similarly?"
3. (Required) Ask students to think: "How can you make this drawing app more interesting to play with others?"

Appendix 3

L3U8.6-8.8 Teacher's Guide: Lesson 3

Teaching Objectives

1. Focus: Guide students in the use of data structures by using CloudDB to synchronize drawing data.
2. Encourage students to work collaboratively to build and test a multiplayer game.

Lesson Outline

Review of Drawing App (5 minutes)

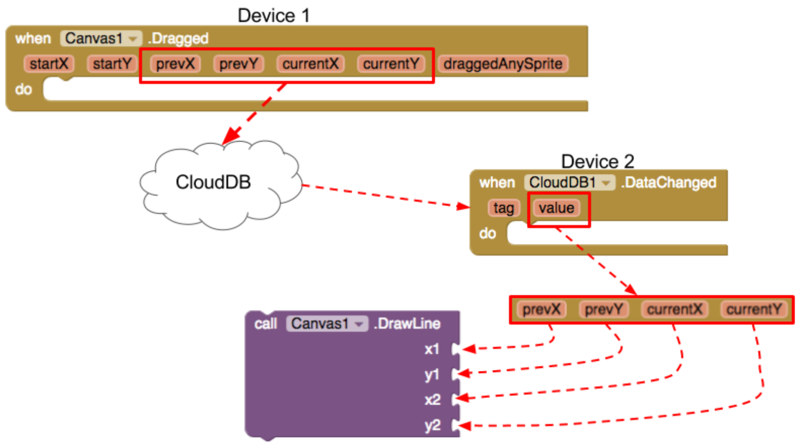
1. Review the drawing app by asking students "How can you play this game with another person on a single device?"
2. Ask students, "How can you play this game with partners at home?"
3. Teacher demonstrates the CloudDB version of the drawing app to show Sketch and Guess working between two devices.

Demonstration of Communication with CloudDB (10 minutes)

1. Ask students, "If we want to send drawing information between devices, what information do we need to send?" (Some students should answer, the x and y coordinates for the DrawLine command.)
2. Explain that whenever a user draws something on their device, they will also save the x,y coordinates for the start and end of the drag event to CloudDB. Other users will take that information and use it to draw the same line on their device.
3. Run the unplugged activity, *Sketch and Guess Unplugged*, to role-play the sketcher, guesser, and CloudDB to demonstrate how the app will send drawing information from the Sketcher to the Guessers.

Adding CloudDB Component to the App (15 minutes)

1. Review the use of CloudDB in the app to pass drawing information from the Sketcher to the Guesser.



2. Ask students to work in pairs to follow the *Student Guide: Lesson 3*, to add CloudDB to their apps. For those students who missed the previous lesson, or were unable to complete the lesson, provide them with the SAG_MultiScreen_checkpoint2.aia file.

3. Since students are only coding the Sketcher part of storing the information in CloudDB, they won't test the app in this lesson. They will have to complete the Guesser code to receive the data from CloudDB in Lesson 4 before they can test.

Wrap-Up (5 minutes)

1. Ask students what code is needed to allow the other app users to see the drawing on their mobile device. (Students may say that the sketcher has stored the data, but the guesser must get the drawing data from CloudDB.)

2. Ask students if there is any app they used before has CloudDB sharing data between users.

3. Explain that in Lesson 4, students will add the guesser code so the guessers see what is being drawn.

Post-Lesson Activities

1. (Required) Complete code if not finished in class.

2. Ask students to think about how they can code the app so the guesser gets the drawing information from CloudDB.

Appendix 4

L3U8.6-8.8 Teacher's Guide: Lesson 4

Teaching Objectives

1. Teach students how to use nested if blocks, to create more complex conditionals.
2. Focus: Assist students in using CloudDB to send information to different users of the app.
3. Encourage students to work collaboratively to create and test a working app.

Lesson Outline

Review and Introduction to Lesson (5 minutes)

1. Ask students what parts they have coded so far in the app.
   (1) Drawing a picture on the screen.
   (2) Storing the drawing information in CloudDB.
2. Explain that in today's lesson, students will code the GuesserScreen, so the guesser will receive the drawing information from CloudDB.

Coding Activity (25 minutes)

1. Students code the GuesserScreen using the *Student Guide: Lesson 4* so the guesser receives the drawing information from CloudDB. Students will use the **CloudDB.DataChanged** event block to receive information about the drawing from CloudDB.

2. Ask students to test the app with their partner. Explain to students that they will use the Build menu to build the apk file, instead of using the **Connect** menu and the MIT AI2 Companion. **Both students will scan the same QR code to ensure they are playing the same app.** <u>Students should decide beforehand who will draw and who will guess, and press the appropriate button on Screen1 when testing.</u>

Wrap-Up (5 minutes)

1. Ask students "What is still missing from the app?" (Students might suggest that they still need to write code so the Guesser can guess what is being drawn.)
2. Explain to students that they will implement functionality for guessing in Lesson 5.
3. Review computational thinking concepts covered in this lesson.

Post-Lesson Activities

(Required) Complete code if not finished in class.

Appendix 5

L3U8.6-8.8 Teacher's Guide: Lesson 5

Teaching Objectives

1. Focus: Assist students in using multiple CloudDB tags and variables to share and keep track of what is being drawn in the app.
2. Encourage students to work collaboratively to create and test a working app.

Lesson Outline

Review and Introduction to Lesson (5 minutes)

1. Review the functions for the Sketcher and the Guesser in the app.
2. Explain to students that they will code the first step in adding guessing functionality in the app. They will do this by adding a second tag for storing the object being drawn in CloudDB. The object that is being drawn by the Sketcher will need to be stored in CloudDB and sent to the Guesser.

Coding the Guessing Functionality (25 minutes)

Following the *Student Guide: Lesson 5*, students add code to store the CurrentDrawing in CloudDB the app. This is just the first step in adding guessing functionality. Students do not have to test their code. They will test with the added code in Lesson 6 to check for a correct answer by the Guesser.

Wrap-up (5 minutes)

1. Ask students, "What is still missing from the app?"
   (1) Students might say the Guesser needs a way to guess what is being drawn.
2. Explain to students that they will add an answer checking in the next lesson.
3. Review computational thinking concepts covered in this lesson.

Post-Lesson Activities

(Required) Complete the *Student Guide: Lesson 5* if not finished.

Appendix 6

L3U8.6-8.8 Teacher's Guide: Lesson 6

Teaching Objectives

1. Introduce the Spinner component as a way to select from a list of options.
2. Focus: Review the use of conditionals to test for a correct answer in an app.
3. Guide students in being incremental and iterative in adding and testing new features in an app.
4. Encourage students to work collaboratively to improve their apps.

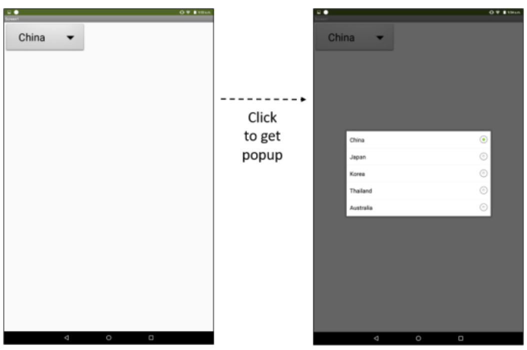
Lesson Outline

Introduction to Lesson (5 minutes)

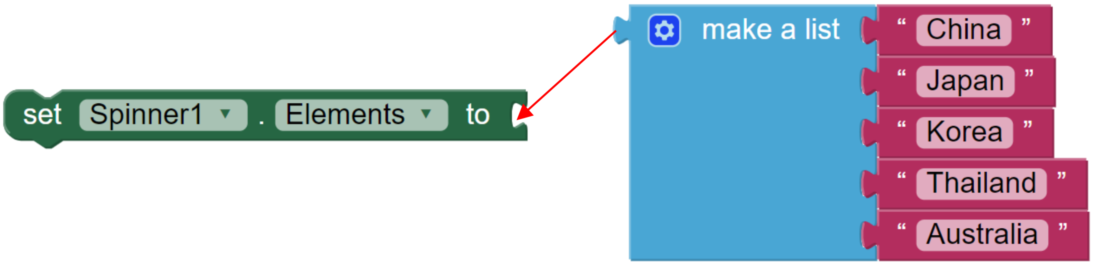
1. Ask students what is missing from the existing Sketch and Guess app.
2. Teacher demonstrates the app with guess checking (SAG_MultiScreen_complete.aia)
3. Explain to students that they will implement guess checking in this lesson by adding code to provide feedback to users when they guess. They will use the CurrentDrawing information stored in CloudDB to test whether the Guesser has made a correct guess or not.

Coding New Features (20 minutes)

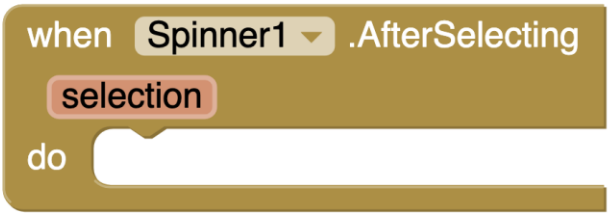
1. Explain to students that they will use the Spinner component, which was a component introduced in the Level 2 Final Project.



You can set the Elements property of the Spinner component to set what gets displayed in the Spinner component list.



The AfterSelected event happens after the user selects an item in the Spinner. The Spinner.Selection is the item selected.



2. Following the *Student Guide: Lesson 6*, students add code to their apps so the Guesser can guess what is being drawn, and the app will let them know if their guess is correct or not. For those students who missed the previous lesson, or were unable to complete the lesson, provide them with the SAG_MultiScreen_checkpoint5.aia file.

3. Student groups test the app for the new functionality. Remind students they will test by building the apk file from the Build menu, and both students scanning the same QR Code on their individual tablets. Students should decide beforehand who will be the Sketcher and who will be the Guesser, and press the appropriate button in the app when testing.

4. Any student pairs who complete Lesson 6 coding may attempt the *Student Guide: Challenge* where they can add colour buttons and a slider to change the line width in the app.

Wrap-up (10 minutes)

1. Review the use of CloudDB in this unit.
2. Review computational thinking concepts covered in this unit.
3. Review computational thinking practices used in this unit.

Post-Lesson Activities

1. (Required) Answer online multiple-choice questions.
2. (Required) Complete survey of learning attitudes.
3. (Optional) Add new functionality to app (countdown timers, scoring).