Section 1: Video

https://drive.google.com/file/d/1b6m_P_mcPfPEIsnlXSdVssWiFnaN0bXK/view?usp=sharing

Section 2: Written Response

2a. Provide a written response or audio narration in your video that:

- Identifies the programming language;
- Identifies the purpose of your program; and
- Explains what the video illustrates

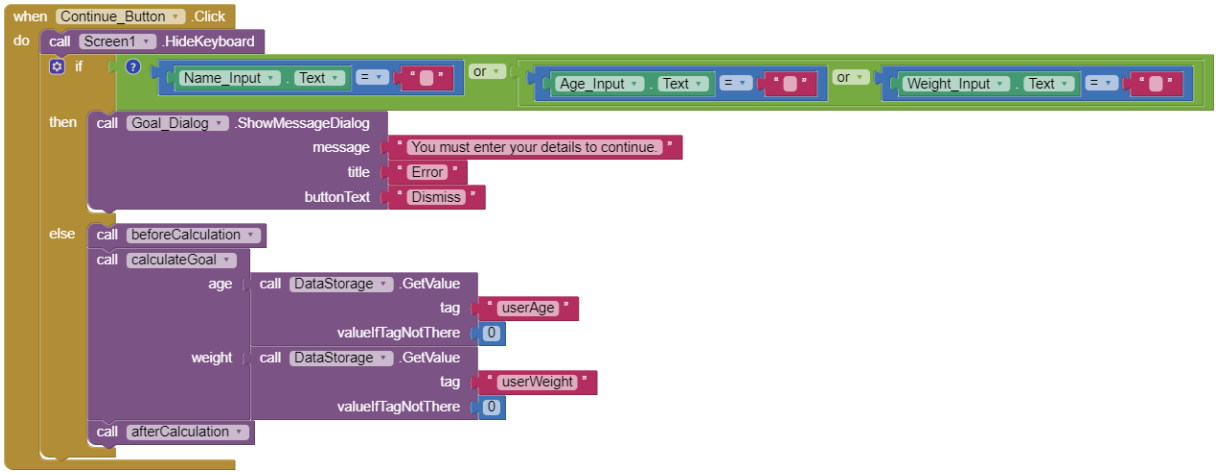
(Approx. 150 words)

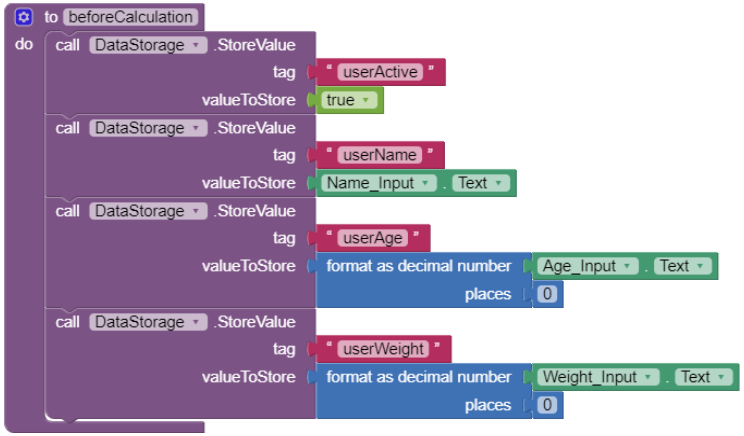
This app, called Water Logged, is a digital journal that gives users the ability to track the number of cups of water they drink each day. Using a mathematical equation with variables based on the user's age and weight, the app prescribes a daily intake goal, and visualizes the user's progress each day. Users can tap the arrow buttons to create a new page and navigate between previous journal entries. The app saves the user's name, age, weight, intake goal, and daily progress as persistent data in a locally stored database. Both the app's user interface and code were developed in the Blockly programming language, facilitated by the App Inventor 2 integrated development environment created by MIT. The short video above shows the user interface of the app and its functionality, demonstrating how the apps progress tracker dynamically responds to variation in the user's input data.

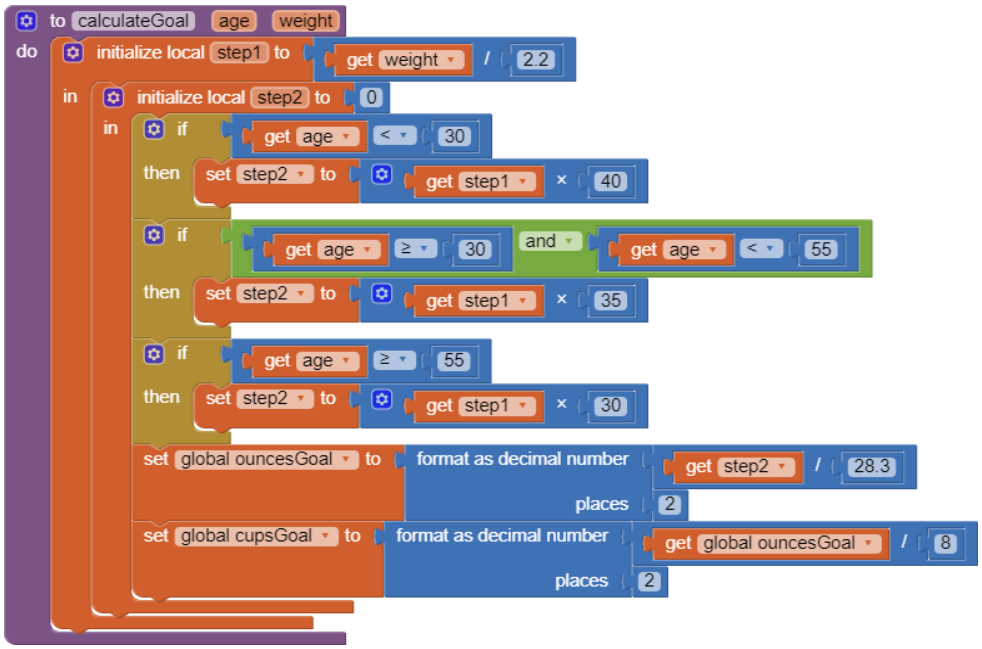
2b. Describe the incremental and iterative development process of your program, focusing on two distinct points in that process. Describe the difficulties and/or opportunities you encountered and how they were resolved or incorporated. In your description clearly indicate whether the development described was collaborative or independent. At least one of these points must refer to independent program development. (Approx. 200 words)

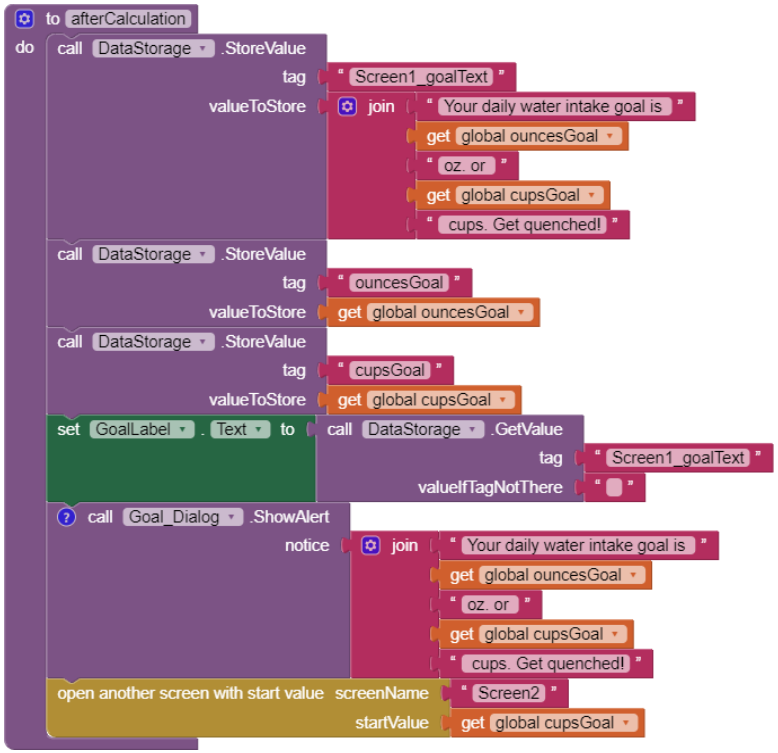
This app was developed entirely independently. It includes code from an app I previously developed for the first Create Performance Task, which facilitated the navigation between journal entries and refreshed or cleared the onscreen data using the displayStoredData and clearStoredData procedures. The concept for this app came to me when I reflected on a discussion I had with my students about apps that they would find useful. One suggestion was an app that would keep track of the amount of water they drank each day, and I decided that this was a challenge I was interested in undertaking. I encountered two major challenges which consumed most of my time while developing the app. The first was that I wanted to be able to show a series of pictures that depicted a glass filling with water as the user advanced in their progress. Finding this series of frames without creating them myself was difficult, but I located a YouTube video tutorial for creating an animation of a glass filling up in Adobe After Effects. As the tutorial animation rolled, I took screenshots of each frame, and altered the photos in Photoshop to remove the stream filling the glass. The second problem I encountered was that I needed to figure out how to update the image when the user reached a certain percentage of their daily progress. Figuring out which variables to plug in to the admittedly simple math equation proved time-consuming, and I was well-served by occasional breaks from the computer in order to approach the problem with fresh eyes later.

2c. Capture and paste the program code segment that implements an algorithm (marked with an oval in section 3 below) that is fundamental for your program to achieve its intended purpose. Your code segment must include an algorithm that integrates other algorithms and integrates mathematical and/or logical concepts. Describe how each algorithm within your selected algorithm functions independently, as well as in combination with others, to form a new algorithm that helps to achieve the intended purpose of the program. (Approx. 200 words)


when Continue_Button .Click
do call Screen1 .HideKeyboard
if Name_Input . Text = "" or Age_Input . Text = "" or Weight_Input . Text = ""
then call Goal_Dialog .ShowMessageDialog
message "You must enter your details to continue."
title "Error"
buttonText "Dismiss"
else call beforeCalculation
call calculateGoal
age call DataStorage .GetValue
tag "userAge"
valueIfTagNotThere 0
weight call DataStorage .GetValue
tag "userWeight"
valueIfTagNotThere 0
call afterCalculation


to beforeCalculation
do call DataStorage .StoreValue
tag "userActive"
valueToStore true
call DataStorage .StoreValue
tag "userName"
valueToStore Name_Input . Text
call DataStorage .StoreValue
tag "userAge"
valueToStore format as decimal number Age_Input . Text
places 0
call DataStorage .StoreValue
tag "userWeight"
valueToStore format as decimal number Weight_Input . Text
places 0


to calculateGoal age weight
do initialize local step1 to get weight / 2.2
in initialize local step2 to 0
in if get age < 30
then set step2 to get step1 × 40
if get age ≥ 30 and get age < 55
then set step2 to get step1 × 35
if get age ≥ 55
then set step2 to get step1 × 30
set global ouncesGoal to format as decimal number get step2 / 28.3
places 2
set global cupsGoal to format as decimal number get global ouncesGoal / 8
places 2


to afterCalculation
do call DataStorage .StoreValue
tag "Screen1_goalText"
valueToStore join "Your daily water intake goal is"
get global ouncesGoal
"oz. or"
get global cupsGoal
"cups. Get quenched!"
call DataStorage .StoreValue
tag "ouncesGoal"
valueToStore get global ouncesGoal
call DataStorage .StoreValue
tag "cupsGoal"
valueToStore get global cupsGoal
set GoalLabel . Text to call DataStorage .GetValue
tag "Screen1_goalText"
valueIfTagNotThere ""
call Goal_Dialog .ShowAlert
notice join "Your daily water intake goal is"
get global ouncesGoal
"oz. or"
get global cupsGoal
"cups. Get quenched!"
open another screen with start value screenName "Screen2"
startValue get global cupsGoal

This code runs when the user presses the "Continue" button on the data entry screen. This screen initializes on the user's first launch of the app after installation, or when the user presses the settings button on the journal screen. This algorithm actually contains four separate sub-algorithms:

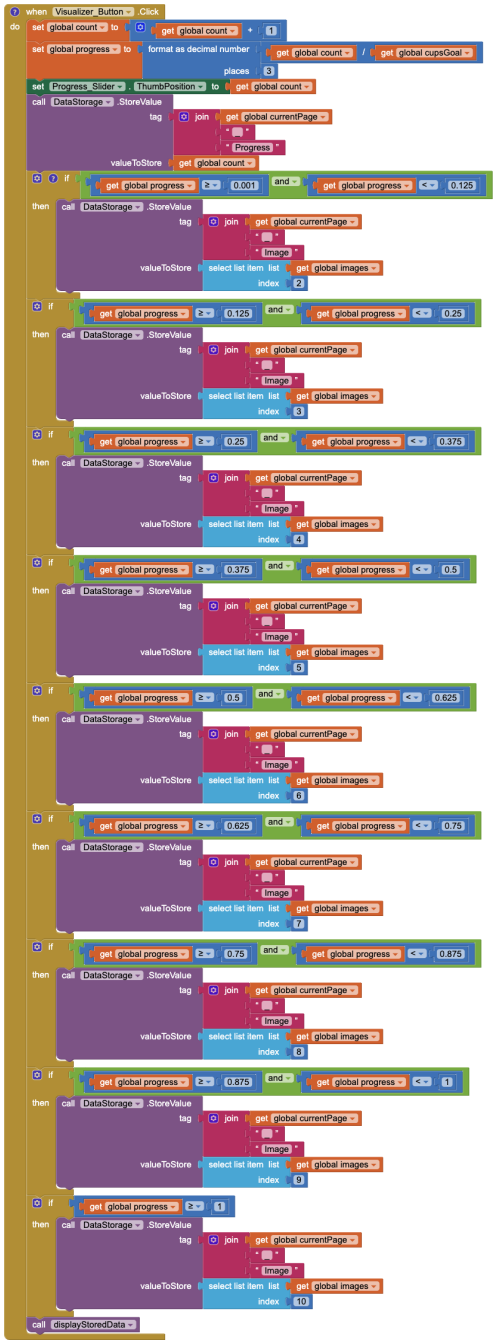
1. After the button is pressed and the keyboard is hidden, a selection statement determines whether any of the input fields contain a null value. If so, the program will not be able to calculate a daily goal, so the algorithm cannot be allowed to proceed. The result of the condition evaluating to "true" is a dialog prompting the user to fix the error.

If there are no null values, the selection statement evaluates to "false" and the else algorithm is run. This algorithm is split into the other three sub-algorithms, each assigned to its own procedure clearly delineated their start and end points:

2. In the beforeCalculation procedure, The user's input data is stored in a local database so that it will persist through multiple uses of the app. The userName value must also be passed to the next screen, and this can only be done by retrieving it from the database. The userActive value exists so that if user data has already been input and the app is closed, it will skip this screen when it is reopened. This is checked in the "when Screen1.intialize" event.
3. The calculateGoal procedure is the start of another sub-algorithm, which incorporates mathematical and logical concepts. This procedure requires the user's age and weight values, which are retrieved from the database. Based on this data, it evaluates an equation which determines the ideal daily water intake for an individual (in ounces). The equation begins by dividing the user's weight by 2.2. The result (stored as the local variable step1) must be multiplied by either 40, 35, or 30 depending on the user's age. That result (stored as the local variable step2) must be divided by 28.3 to reveal the number of ounces the user should drink each day (stored as ouncesGoal). This value is converted to cups by dividing by 8.
4. The final sub-algorithm is performed in the afterCalculation procedure, and uses the results of the previous sub-algorithm to store the goal values in the database, and then constructing a text string notifying the user of their goal.

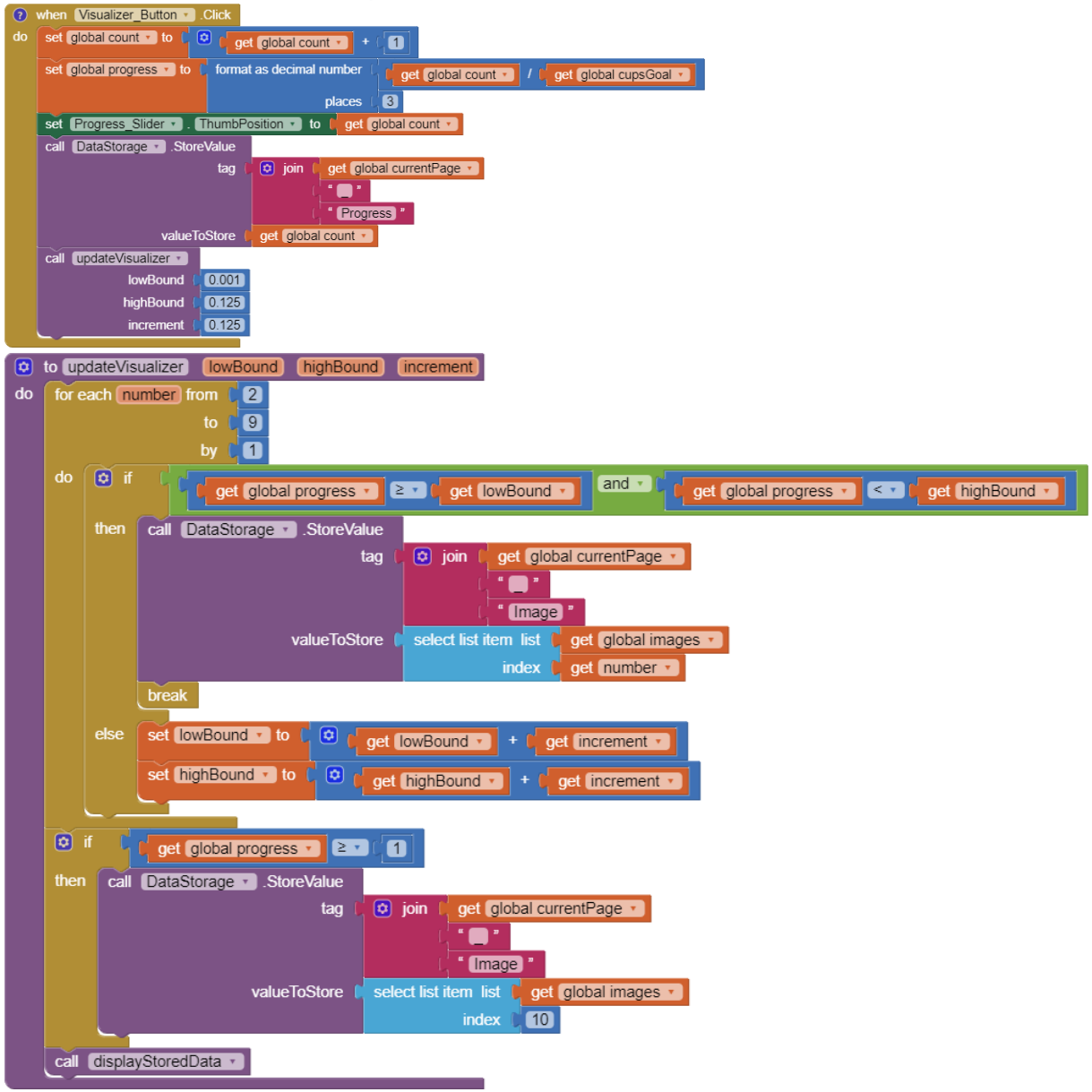
Only after all of the above has been accomplished will the program open the next screen. As it does so, the user's goal is the only datum that is passed directly to Screen2 as input to be stored as a new global variable on that screen. Each of the above algorithms accomplishes its own specific task, but they each cannot succeed without the previous one, stemming from the user's input data as the catalyst.

2d. Capture and paste the program code segment that contains an abstraction you developed (marked with a rectangle in section 3 below). Your abstraction should integrate mathematical and logical concepts. Explain how your abstraction helped manage the complexity of your program. (Approx. 200 words)



Old code

New code

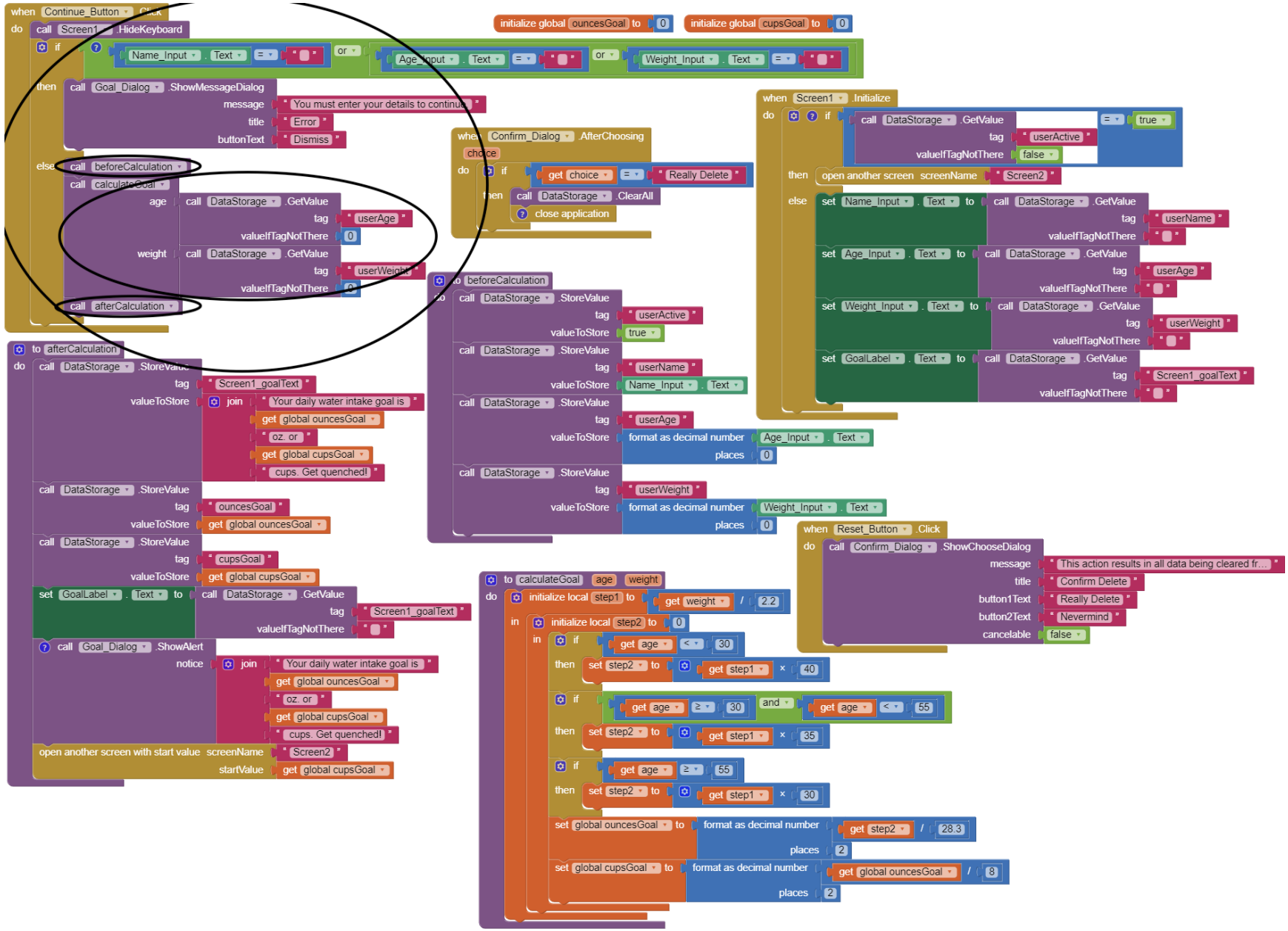


When the user touches the image of the glass to increase the number of cups they have consumed, the Visualizer_Button.Click event is activated. On the left is the original code for this algorithm which hard-coded numerical values and selection statements in order to determine what image to display based on the user's current progress (as a percentage of the goal). Understanding that this algorithm essentially repeated the same process several times, I contemplated a way to abstract this algorithm using a repetition structure. On the right is the resultant algorithm, which consists of an abstraction in the form of the updateVisualizer procedure. The local input variables of this procedure are used in mathematical and logical statements which could potentially result in different values each time through the loop depending on the user's goal and current progress. The number variable in the loop corresponds to the index of the list of images, so it must begin at 2 (which is the first image that shows any progress) and end at 9 (which is the last image that requires the progress percentage to fall *between* a high and low value). The first input variable is the lowBound. Each time through the loop, the program checks whether the progress percentage is greater than or equal to this value. The program also checks whether the progress is less than the highBound input value. The increment input variable is the amount that is added to the high and low bounds each time through the loop if the progress does not fall within the bounds. If at any point the progress does fall within the bounds, the current number is passed as the index to the images list, and the loop is broken. If the progress meets or exceeds 1, the final image will be displayed. The first attempt at the algorithm required approximately 160 blocks, and the more abstract procedure requires only 43. This is a dramatic improvement which leverages the device's processing capabilities for better program efficiency, and reduces the complexity of the code by mitigating the potential for errors in the development and debugging process.

Section 3: Program Code

Capture and paste your entire program code in this section.

- Mark with an oval the segment of program code that implements the algorithm you created for your program that integrates other algorithms and integrates mathematical and/or logical concepts.
- Mark with a rectangle the segment of program code that represents an abstraction you developed.
- Include comments or citations for program code that has been written by someone else.


when Continue_Button .Click
do call Screen1 .HideKeyboard
if Name_Input . Text = "" or Age_Input . Text = "" or Weight_Input . Text = ""
then call Goal_Dialog .ShowMessageDialog
message "You must enter your details to continue"
title "Error"
buttonText "Dismiss"
else call beforeCalculation
call calculateGoal
age call DataStorage .GetValue tag "userAge" valueIfTagNotThere 0
weight call DataStorage .GetValue tag "userWeight" valueIfTagNotThere 0
call afterCalculation
initialize global ouncesGoal to 0
initialize global cupsGoal to 0
when Confirm_Dialog .AfterChoosing
choice
do if get choice = "Really Delete"
then call DataStorage .ClearAll
close application
when Screen1 .Initialize
do if call DataStorage .GetValue tag "userActive" valueIfTagNotThere false = true
then open another screen screenName "Screen2"
else set Name_Input . Text to call DataStorage .GetValue tag "userName" valueIfTagNotThere ""
set Age_Input . Text to call DataStorage .GetValue tag "userAge" valueIfTagNotThere ""
set Weight_Input . Text to call DataStorage .GetValue tag "userWeight" valueIfTagNotThere ""
set GoalLabel . Text to call DataStorage .GetValue tag "Screen1_goalText" valueIfTagNotThere ""
to beforeCalculation
do call DataStorage .StoreValue tag "userActive" valueToStore true
call DataStorage .StoreValue tag "userName" valueToStore Name_Input . Text
call DataStorage .StoreValue tag "userAge" valueToStore format as decimal number Age_Input . Text places 0
call DataStorage .StoreValue tag "userWeight" valueToStore format as decimal number Weight_Input . Text places 0
to afterCalculation
do call DataStorage .StoreValue tag "Screen1_goalText" valueToStore join "Your daily water intake goal is" get global ouncesGoal "oz. or" get global cupsGoal "cups. Get quenched!"
call DataStorage .StoreValue tag "ouncesGoal" valueToStore get global ouncesGoal
call DataStorage .StoreValue tag "cupsGoal" valueToStore get global cupsGoal
set GoalLabel . Text to call DataStorage .GetValue tag "Screen1_goalText" valueIfTagNotThere ""
call Goal_Dialog .ShowAlert
notice join "Your daily water intake goal is" get global ouncesGoal "oz. or" get global cupsGoal "cups. Get quenched!"
open another screen with start value screenName "Screen2" startValue get global cupsGoal
when Reset_Button .Click
do call Confirm_Dialog .ShowChooseDialog
message "This action results in all data being cleared fr..."
title "Confirm Delete"
button1Text "Really Delete"
button2Text "Nevermind"
cancelable false
to calculateGoal age weight
do initialize local step1 to get weight / 2.2
in initialize local step2 to 0
in if get age < 30
then set step2 to get step1 × 40
if get age ≥ 30 and get age < 55
then set step2 to get step1 × 35
if get age ≥ 55
then set step2 to get step1 × 30
set global ouncesGoal to format as decimal number get step2 / 28.3 places 2
set global cupsGoal to format as decimal number get global ouncesGoal / 8 places 2

Screen1 > > >


initialize global cupsGoal to get start value
initialize global currentPage to 1
initialize global totalPages to 1
initialize global count to 0
initialize global progress to 0
initialize global images to make a list
1.png
2.png
3.png
4.png
5.png
6.png
7.png
8.png
9.png
10.png
when Screen2 .Initialize
do call displayStoredData
when Clear_Button .Click
when Forward_Button .Click
when Visualizer_Button .Click
to displayStoredData
to clearStoredData
to updateVisualizer lowBound highBound increment
when Back_Button .Click
when Settings_Button .Click
when Date_Picker .AfterDateSet

Screen2 > > >