ICSP

AppInventor
Lecture 3

Outline

- Lists
- Subroutines
- Communication protocols

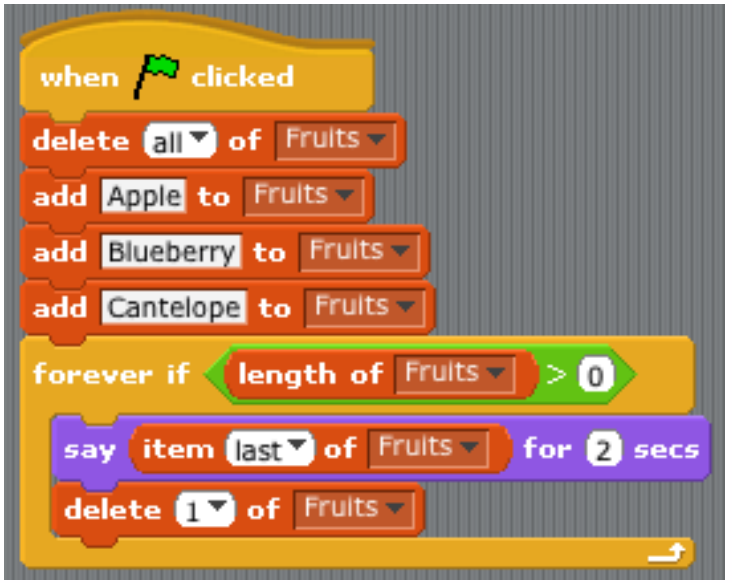
Lists

- Abstract data structure that implements an ordered collection of values, the same value can occur more than once
- The size of the list indicates the number of elements
- Elements can be accessed via their index
- Elements can be concattenated to the list

List:	c	4	a	1
Index:	0	1	2	3

List example

- When the program starts fruits will be added to the fruit list.
- After the fruits are added the Sprite will, as long as there are fruits in the list, say the name of the last fruit in the list and then remove it from the list

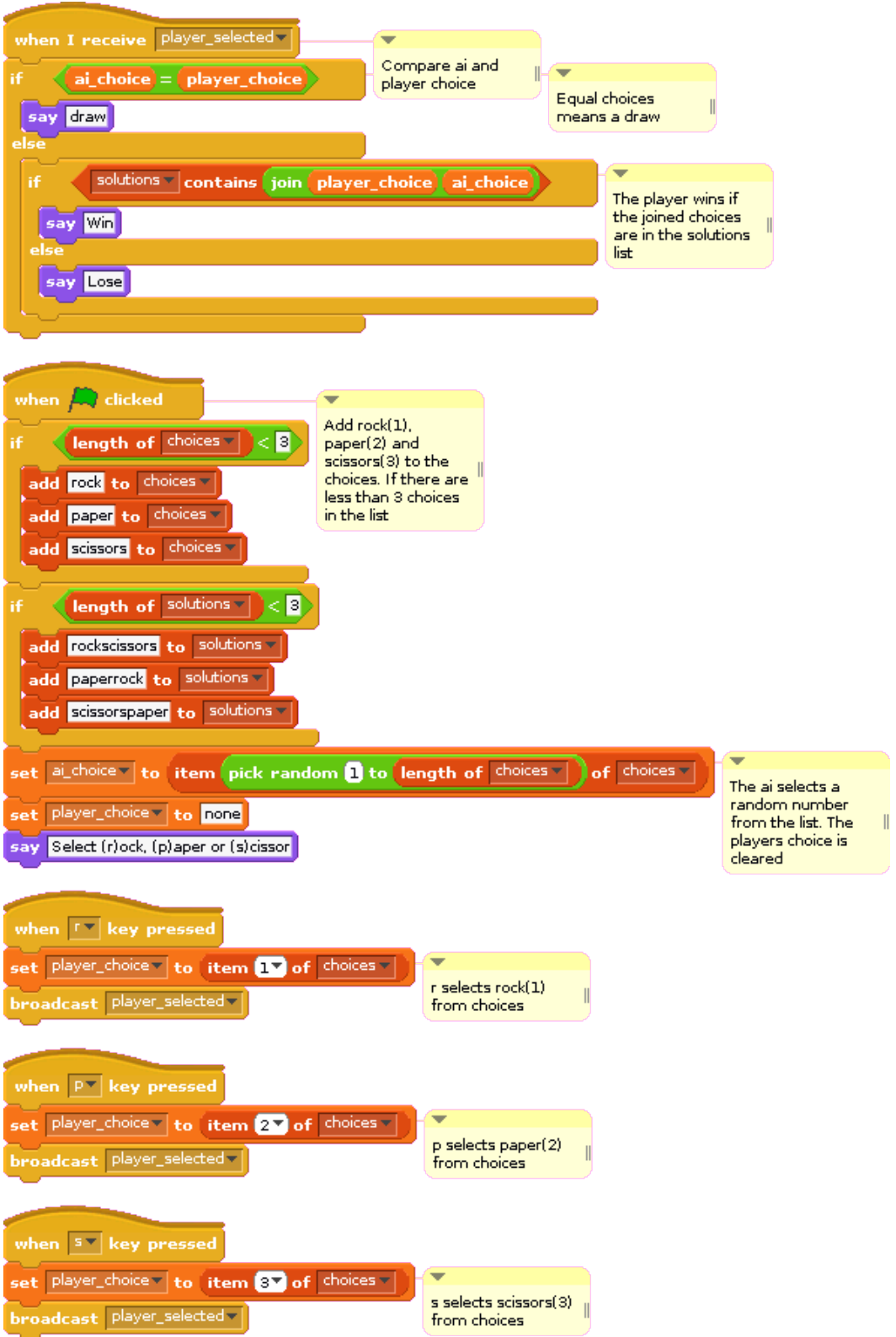


Queues & Stacks

- Queues and Stacks are two versions of lists with different properties
- In a queue the first element that is added to the queue (enqueue) is the first to be taken from it (dequeue) Called First in, First Out (FIFO)
  - E.g. The checkout line in a supermarket is a queue
- In a stack the last element to be put(pushed) on the stack is the first to be taken off the top(popped) Called Last in, First Out (LIFO)
  - E.g. A pile of books can be described as a stack

Rock, Paper or Scissors

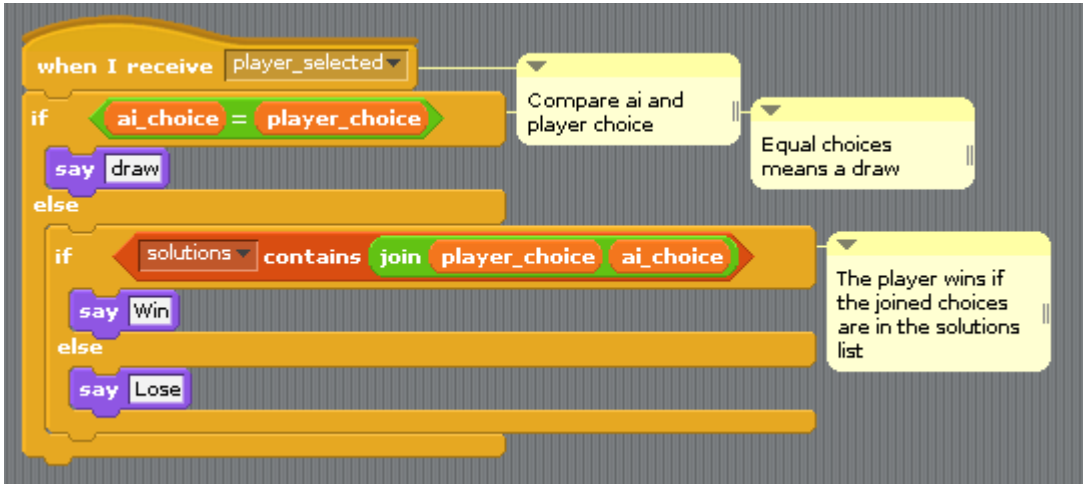
- Rules:
  - Rock beats Scissors
  - Scissors beats Paper
  - Paper beats Rock
- **Exercise**: Code a Rock, Paper or Scissors game in Scratch:
  - You make a choice of Rock, Paper or Scissors with the keys “r”, “p” or “s”
  - The computer makes a random choice Rock, Paper or Scissors
  - The result is displayed
- (Bonus if you use a list, e.g. The values of “rock”, “paper” or “scissors” are all player choices)


when I receive player_selected
if ai_choice = player_choice
say draw
else
if solutions contains join player_choice ai_choice
say Win
else
say Lose
Compare ai and player choice
Equal choices means a draw
The player wins if the joined choices are in the solutions list
when clicked
if length of choices < 3
add rock to choices
add paper to choices
add scissors to choices
Add rock(1), paper(2) and scissors(3) to the choices. If there are less than 3 choices in the list
if length of solutions < 3
add rockscissors to solutions
add paperrock to solutions
add scissorspaper to solutions
set ai_choice to item pick random 1 to length of choices of choices
set player_choice to none
say Select (r)ock, (p)aper or (s)cissor
The ai selects a random number from the list. The players choice is cleared
when r key pressed
set player_choice to item 1 of choices
broadcast player_selected
r selects rock(1) from choices
when p key pressed
set player_choice to item 2 of choices
broadcast player_selected
p selects paper(2) from choices
when s key pressed
set player_choice to item 3 of choices
broadcast player_selected
s selects scissors(3) from choices

Subroutine

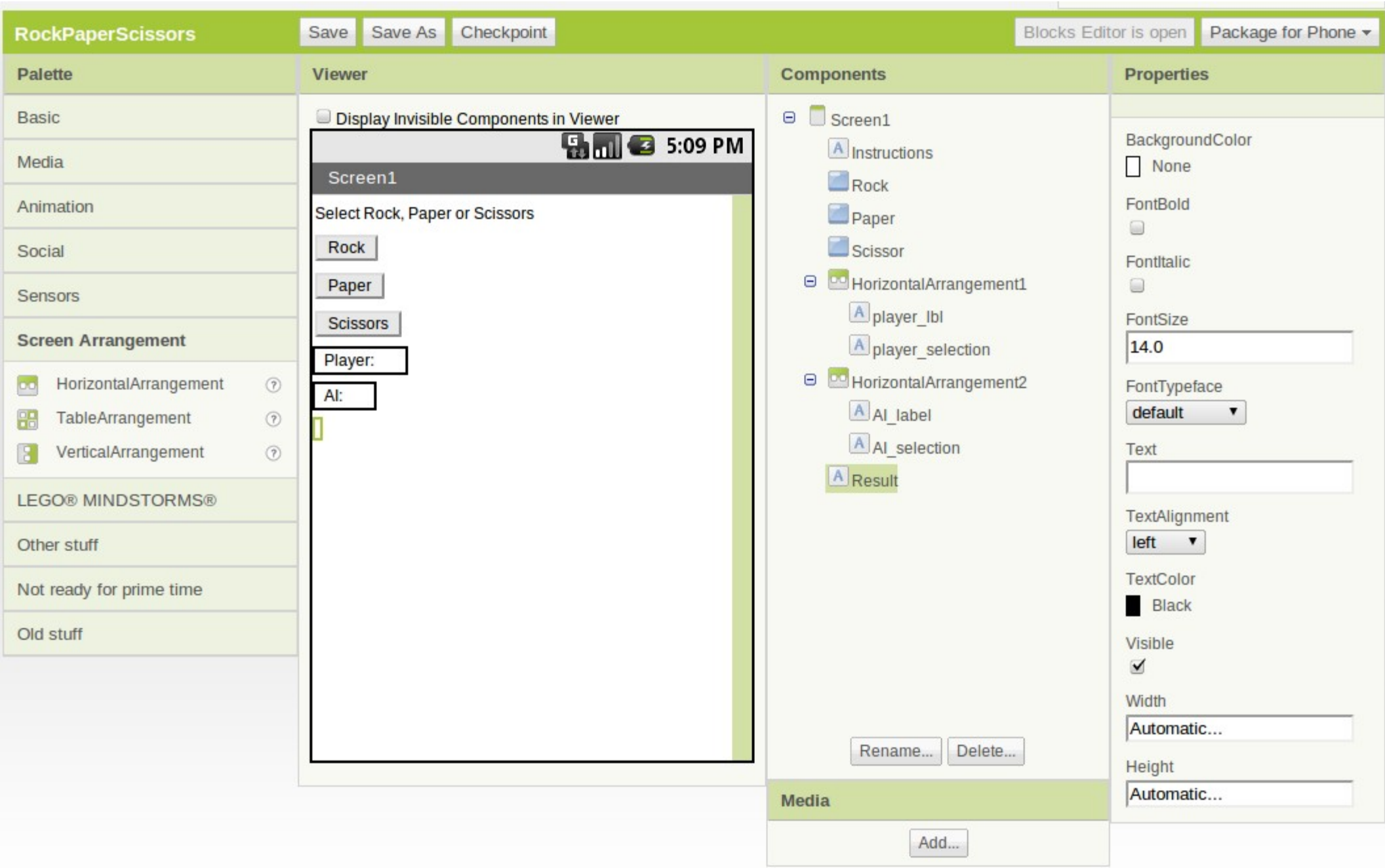
A subroutine (procedure, function, routine, method, or subprogram)

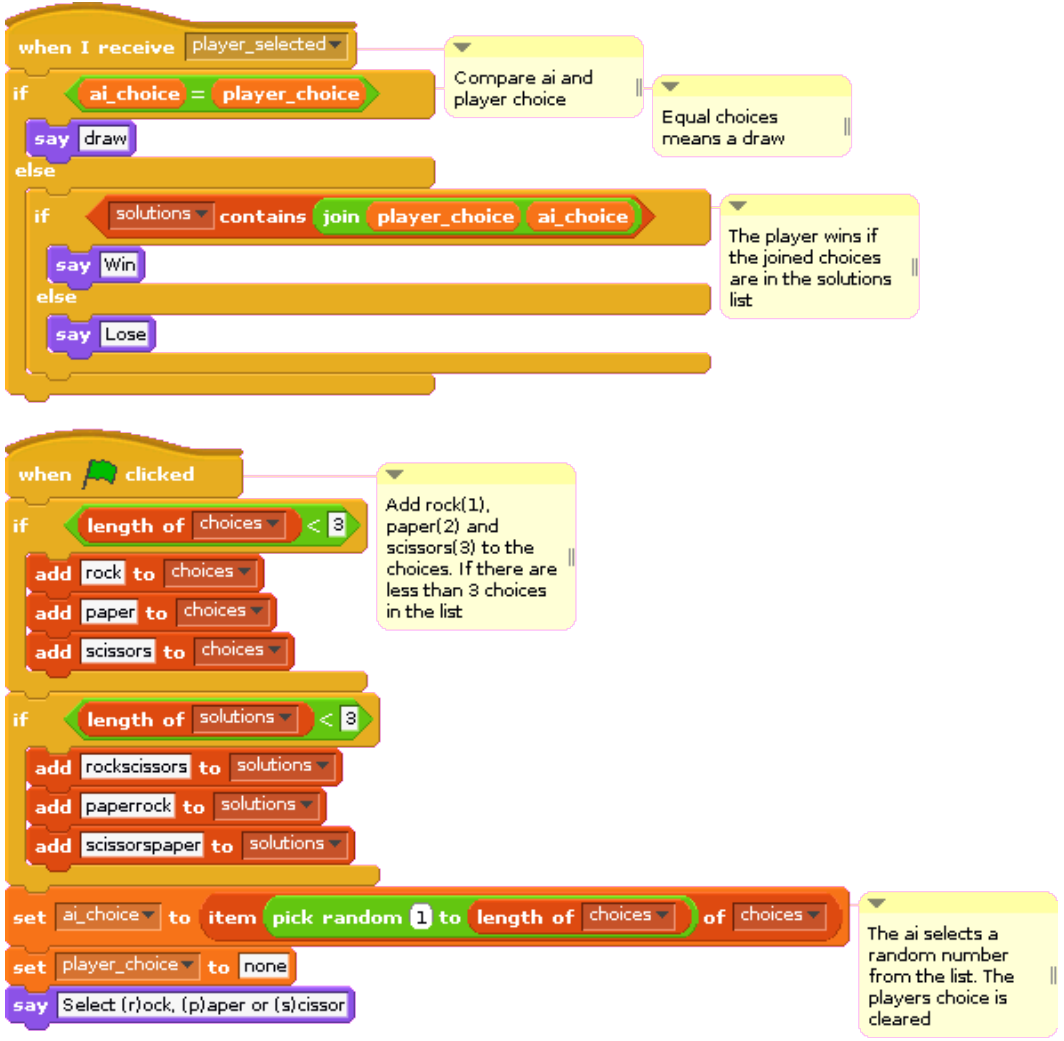
- Portion of code within a larger program that performs a specific task and is somewhat independent piece of code.
  - Used to structure the code
  - Used when repeated calls need to be made
  - Reduces the cost of developing and maintaining a large program and increases its quality and reliability.
- Subroutines, collected into libraries, are an important mechanism for sharing software

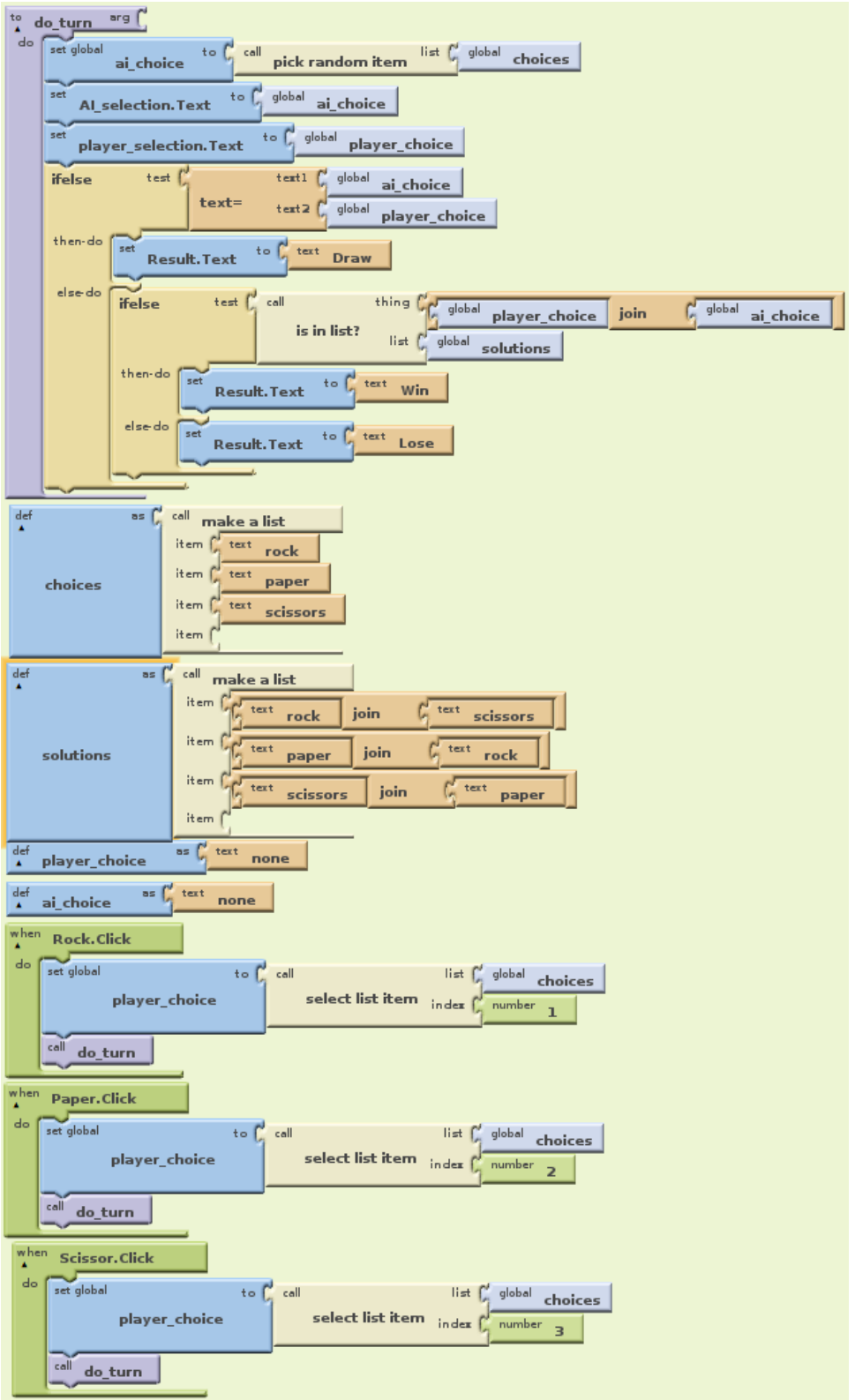


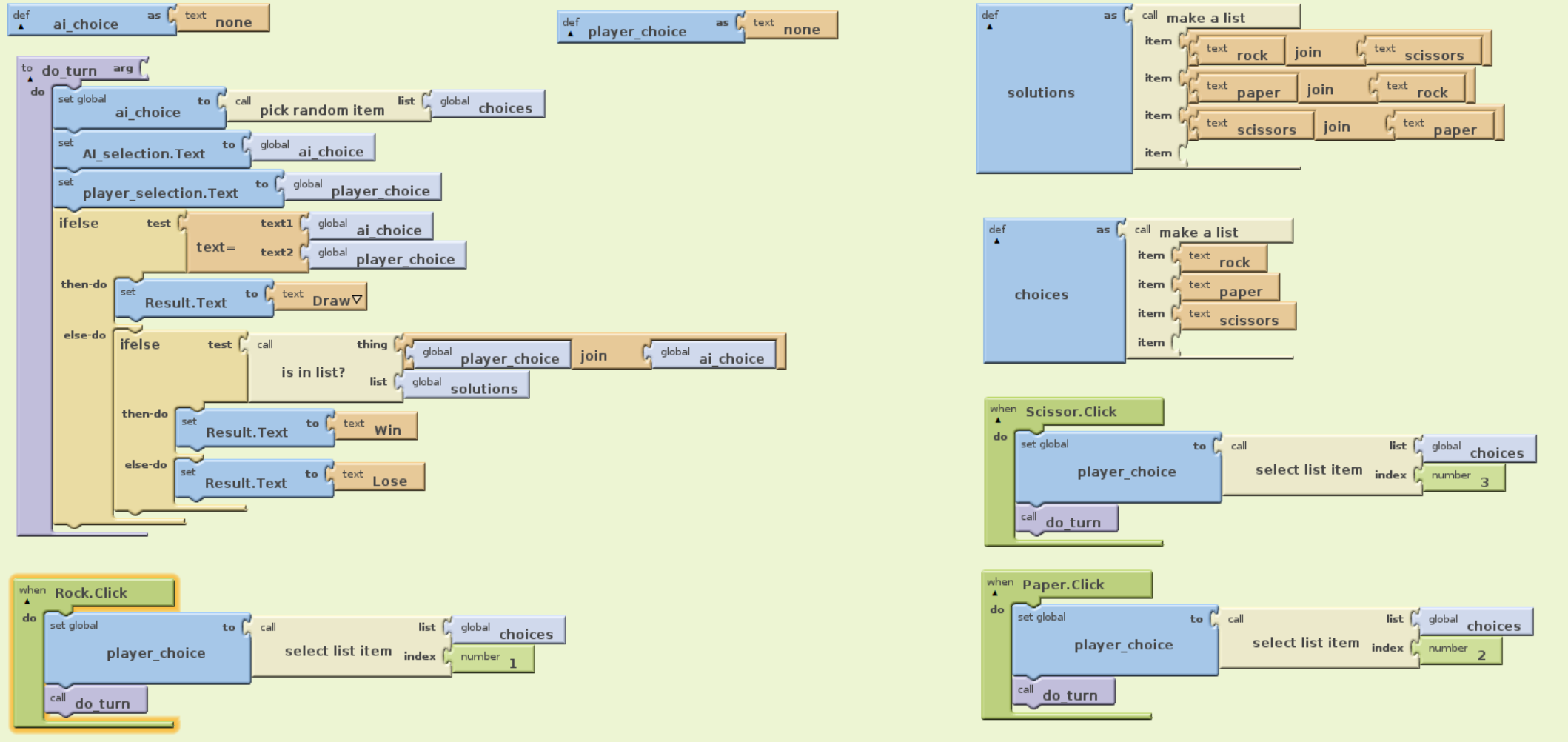
AppInventor Rock, Paper and Scissors

- Create 3 buttons
  - Rock
  - Paper
  - Scissors
- Display the result of your choice against a random AI
- Think from Scratch


RockPaperScissors
Save
Save As
Checkpoint
Blocks Editor is open
Package for Phone
Palette
Basic
Media
Animation
Social
Sensors
Screen Arrangement
HorizontalArrangement
TableArrangement
VerticalArrangement
LEGO® MINDSTORMS®
Other stuff
Not ready for prime time
Old stuff
Viewer
Display Invisible Components in Viewer
5:09 PM
Screen1
Select Rock, Paper or Scissors
Rock
Paper
Scissors
Player:
AI:
Components
Screen1
Instructions
Rock
Paper
Scissor
HorizontalArrangement1
player_lbl
player_selection
HorizontalArrangement2
AI_label
AI_selection
Result
Rename...
Delete...
Media
Add...
Properties
BackgroundColor
None
FontBold
FontItalic
FontSize
14.0
FontTypeface
default
Text
TextAlignment
left
TextColor
Black
Visible
Width
Automatic...
Height
Automatic...


when I receive player_selected
Compare ai and player choice
if ai_choice = player_choice
Equal choices means a draw
say draw
else
if solutions contains join player_choice ai_choice
The player wins if the joined choices are in the solutions list
say Win
else
say Lose
when clicked
Add rock(1), paper(2) and scissors(3) to the choices. If there are less than 3 choices in the list
if length of choices < 3
add rock to choices
add paper to choices
add scissors to choices
if length of solutions < 3
add rockscissors to solutions
add paperrock to solutions
add scissorspaper to solutions
set ai_choice to item pick random 1 to length of choices of choices
The ai selects a random number from the list. The players choice is cleared
set player_choice to none
say Select (r)ock, (p)aper or (s)cissor
when r key pressed
set player_choice to item 1 of choices
broadcast player_selected
r selects rock(1) from choices
when p key pressed
set player_choice to item 2 of choices
broadcast player_selected
p selects paper(2) from choices
when s key pressed
set player_choice to item 3 of choices
broadcast player_selected
s selects scissors(3) from choices

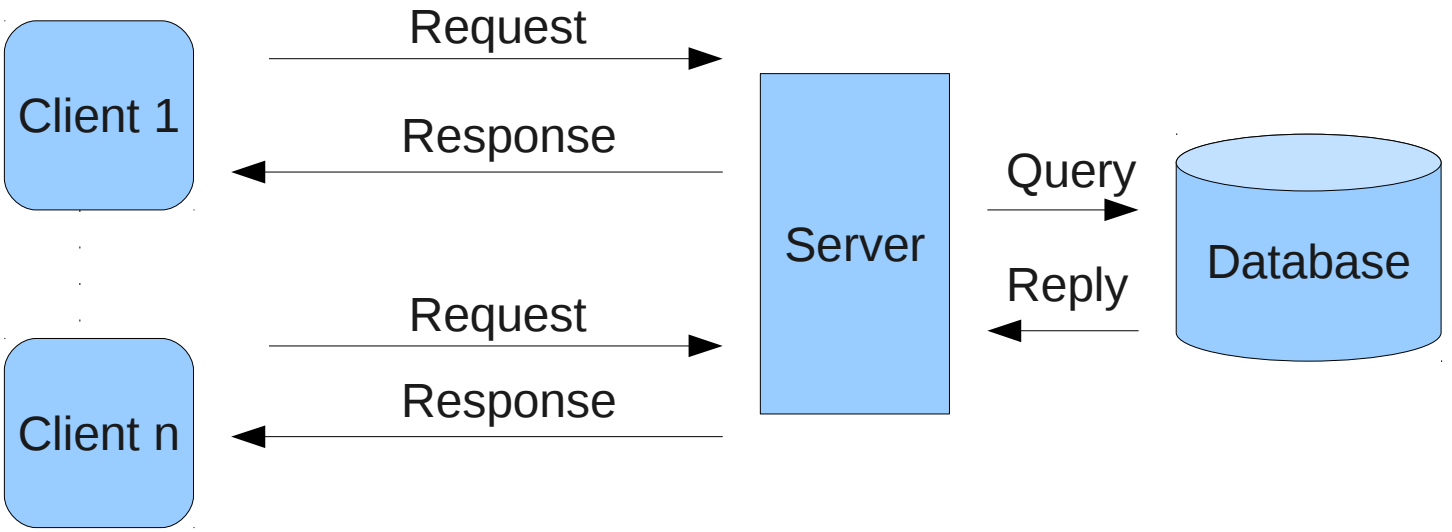

to do_turn arg
do
set global ai_choice to call pick random item list global choices
set AI_selection.Text to global ai_choice
set player_selection.Text to global player_choice
ifelse test text= text1 global ai_choice text2 global player_choice
then-do set Result.Text to text Draw
else-do ifelse test call is in list? thing global player_choice join global ai_choice list global solutions
then-do set Result.Text to text Win
else-do set Result.Text to text Lose
def choices as call make a list item text rock item text paper item text scissors item
def solutions as call make a list item text rock join text scissors item text paper join text rock item text scissors join text paper item
def player_choice as text none
def ai_choice as text none
when Rock.Click
do set global player_choice to call select list item list global choices index number 1
call do_turn
when Paper.Click
do set global player_choice to call select list item list global choices index number 2
call do_turn
when Scissor.Click
do set global player_choice to call select list item list global choices index number 3
call do_turn


def ai_choice as text none
def player_choice as text none
to do_turn arg
do
set global ai_choice to call pick random item list global choices
set AI_selection.Text to global ai_choice
set player_selection.Text to global player_choice
ifelse test text= text1 global ai_choice text2 global player_choice
then-do set Result.Text to text Draw
else-do ifelse test call is in list? thing global player_choice join global ai_choice list global solutions
then-do set Result.Text to text Win
else-do set Result.Text to text Lose
def solutions as call make a list
item text rock join text scissors
item text paper join text rock
item text scissors join text paper
item
def choices as call make a list
item text rock
item text paper
item text scissors
item
when Scissor.Click
do set global player_choice to call select list item list global choices index number 3
call do_turn
when Rock.Click
do set global player_choice to call select list item list global choices index number 1
call do_turn
when Paper.Click
do set global player_choice to call select list item list global choices index number 2
call do_turn

Between a rock, a paper or scissors

- A drama about communication in one act
- Sceen: A Rock, Paper, Scissors tournament
- Actors
  - Clients – Participates in the tournament
    - Name: Rick, likes to roll with Rock
    - Name: Piper, likes to fold with Paper
    - Name: Saussage, likes to run with Scissors
  - Server – Handles the Rock, Paper, Scissors tournament requests, respones and data storage
    - Gets players name and responds with player results
    - Responds with all stored player names if requested
    - Stores player name and move

Communication



Communication Protocols

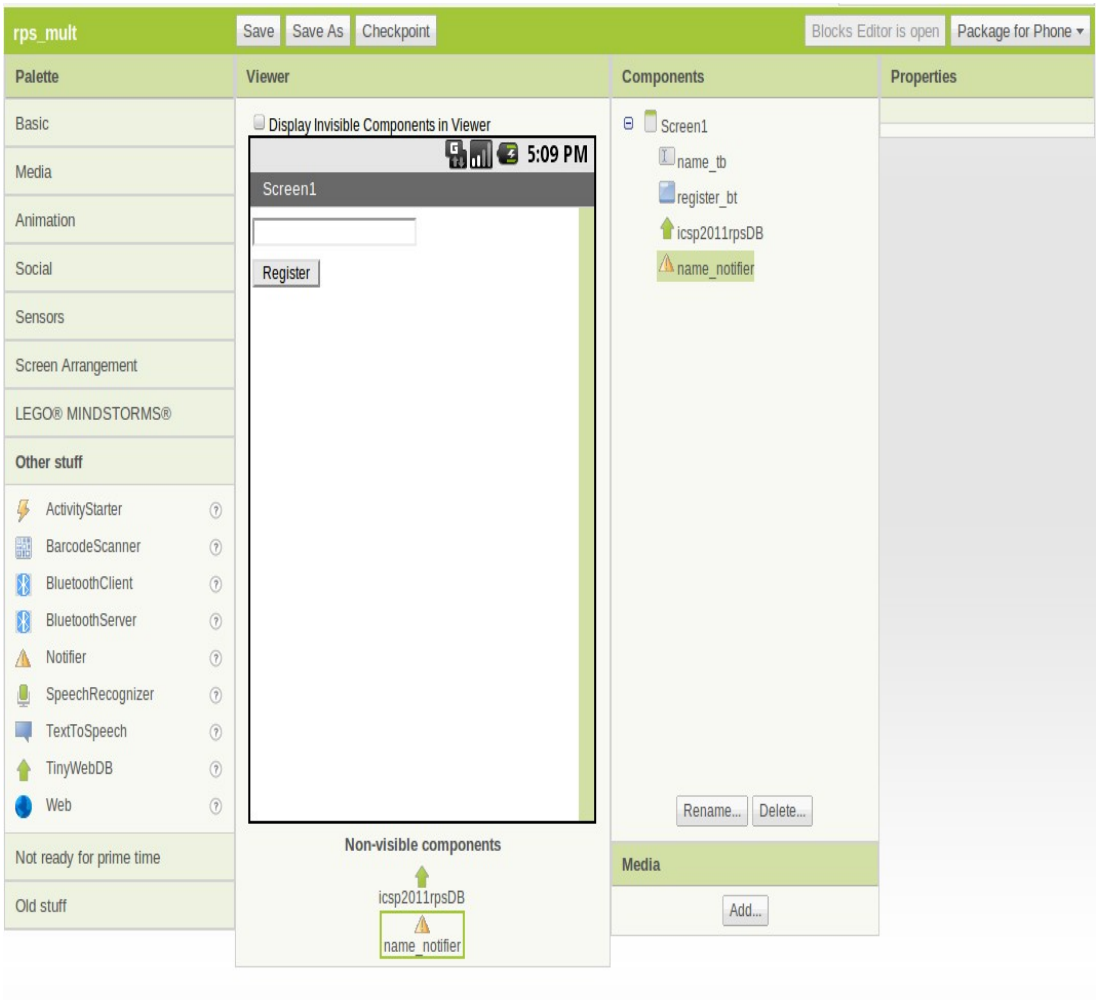
- How do you make the devices (e.g. PC or Smartphone) communicate with each other in order to register Rock, Paper or Scissors players, moves and results?
  - **Excersice**: Write down in pseudocode how you would handle the communication

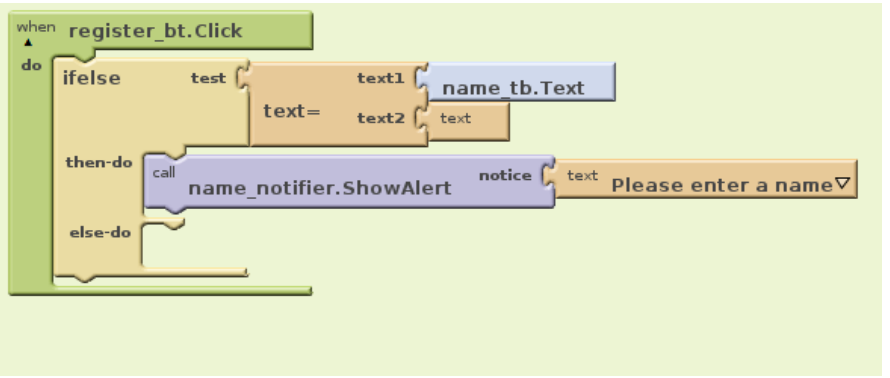
Rock, Paper or Scissors - Multiplayer

- The devices do not know that other devices exist. Therefore the webpage serves the clients(apps) and stores information.
  - A webpage has been set up http://icsp2011rps.appspot.com/

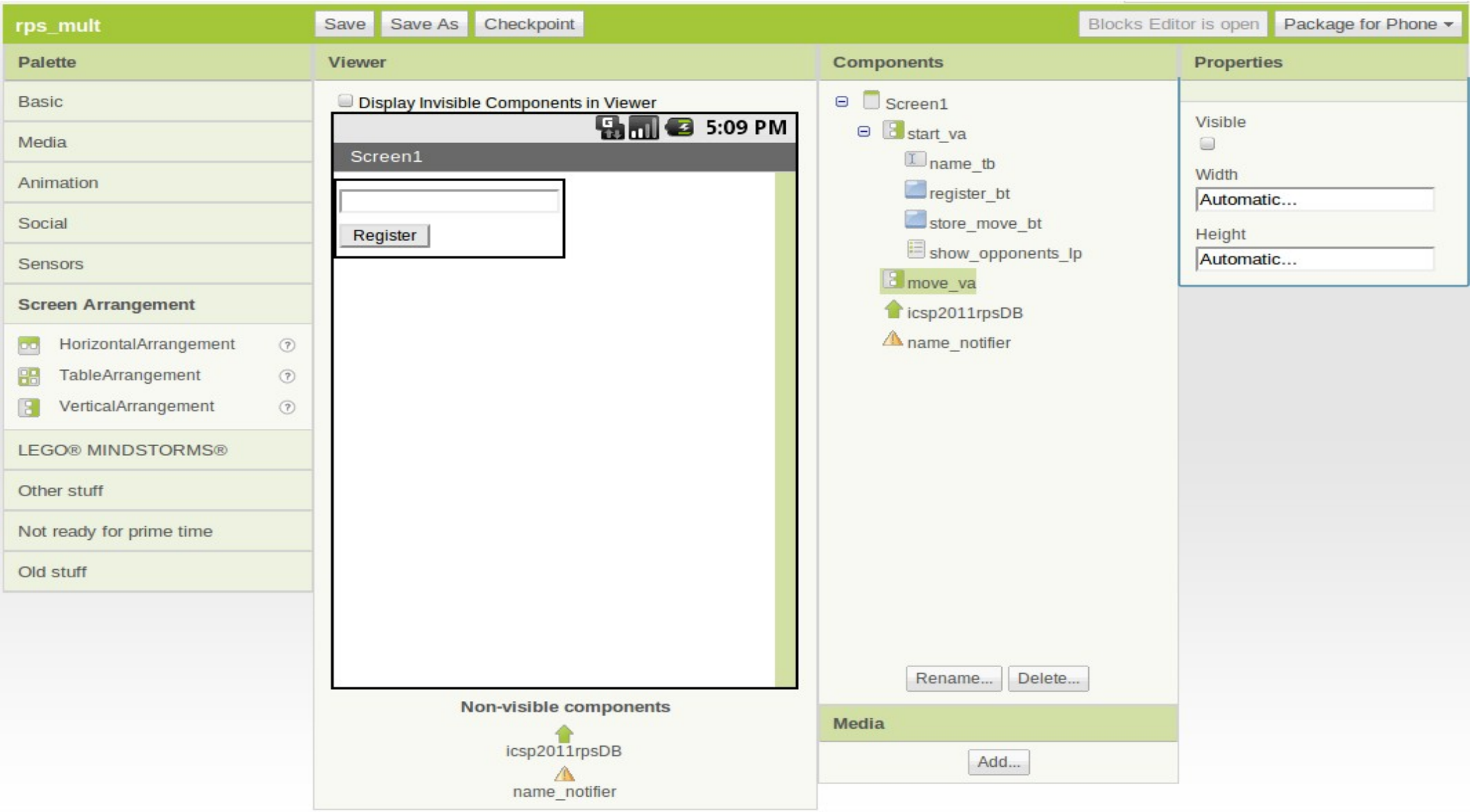
1. Register a name on the device
    1. If there is no name registred notify that a name is missing
    2. Else show the options “Store Move” and “Compete against an opponent”
2. Store a move on the server in the registered name
    1. Play a move
    2. Store the move on the server's data base
3. Compete against an opponent. The moves and opponents are stored on the server
    1. Chose an opponent listed in the server's database
    2. Play a move
    3. Calculate the result
    4. Show the result

Register a name

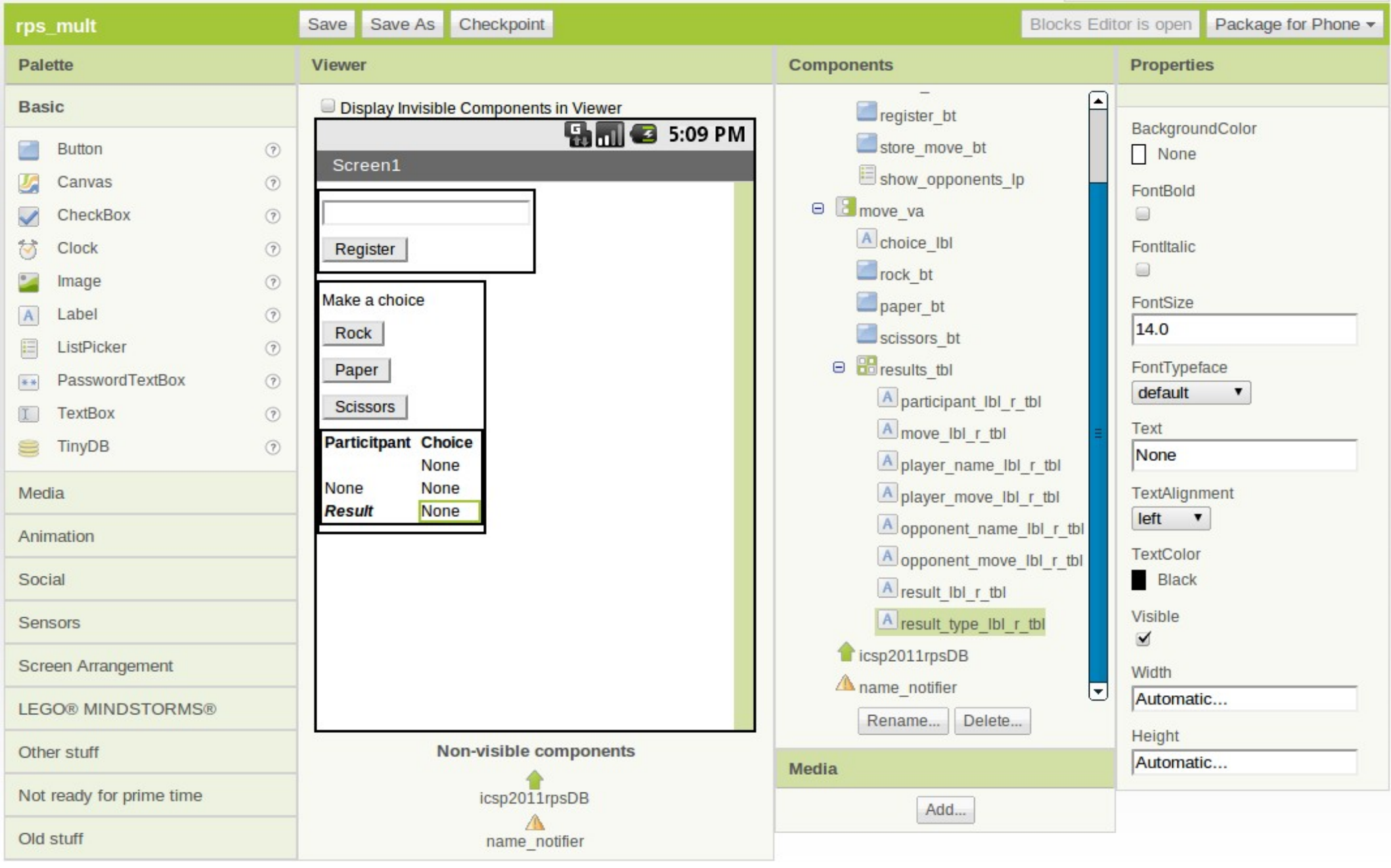




Chose to Store Move or Play Opponent

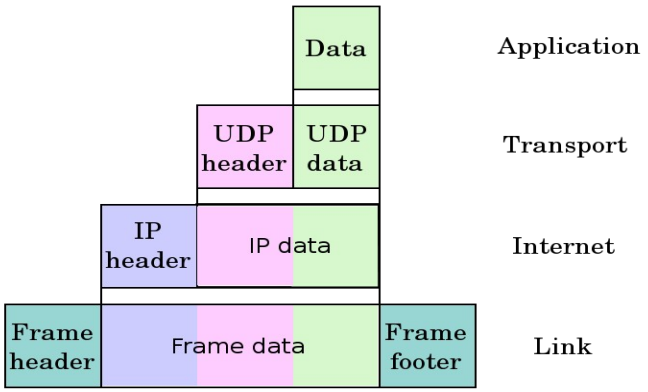


Choose a Move



Internet Protocol Stack

- The lowest protocol deals with physical interaction of the hardware. Every layer above adds more features.



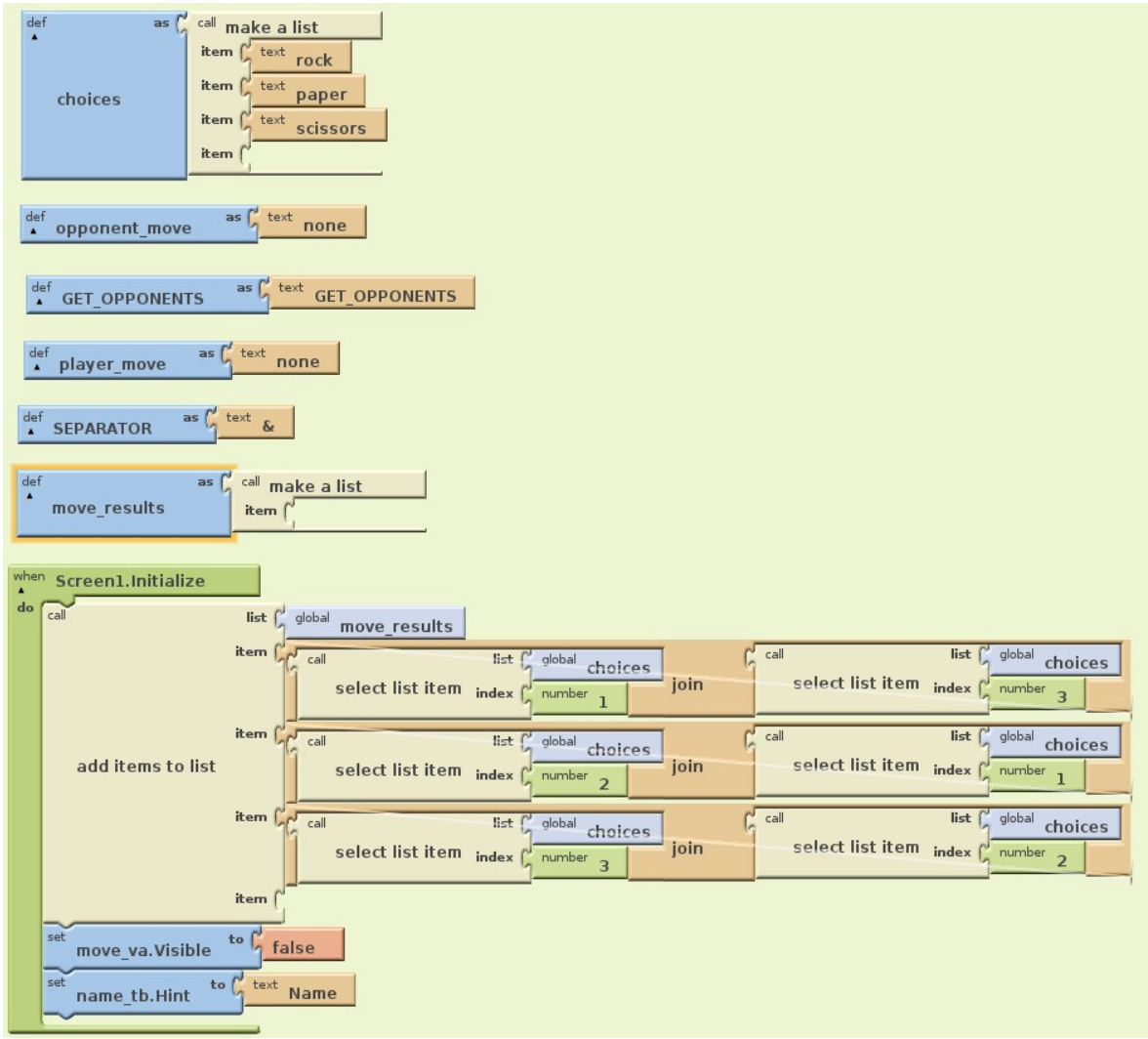
Protocol	Layer
HTTP	Application
TCP	Transport
IP	Internet
Ethernet	Link
IEEE 802.3u	Physical

Protocol specifications

- Requesting to store a player's move, `SetValue`
  - Use the player name as the key
  - Use the choice name as the value
- Requesting a list of opponents, `GetValue`
  - Use the key “GET_OPPONENTS”
  - Query returns a string value of opponent names joined by “&”
  - E.g. Request “GET_OPPONENTS” returns “Rick&Piper”, i.e. Rick and Piper are possible opponents
  - Usefull blocks for handling data transfer is
    - “split”, splits a string at a specific delimiter
- Requesting a player move, `GetValue`
  - Use the player name as the key
  - E.g. Request “Rick” returns the value of Rick's move

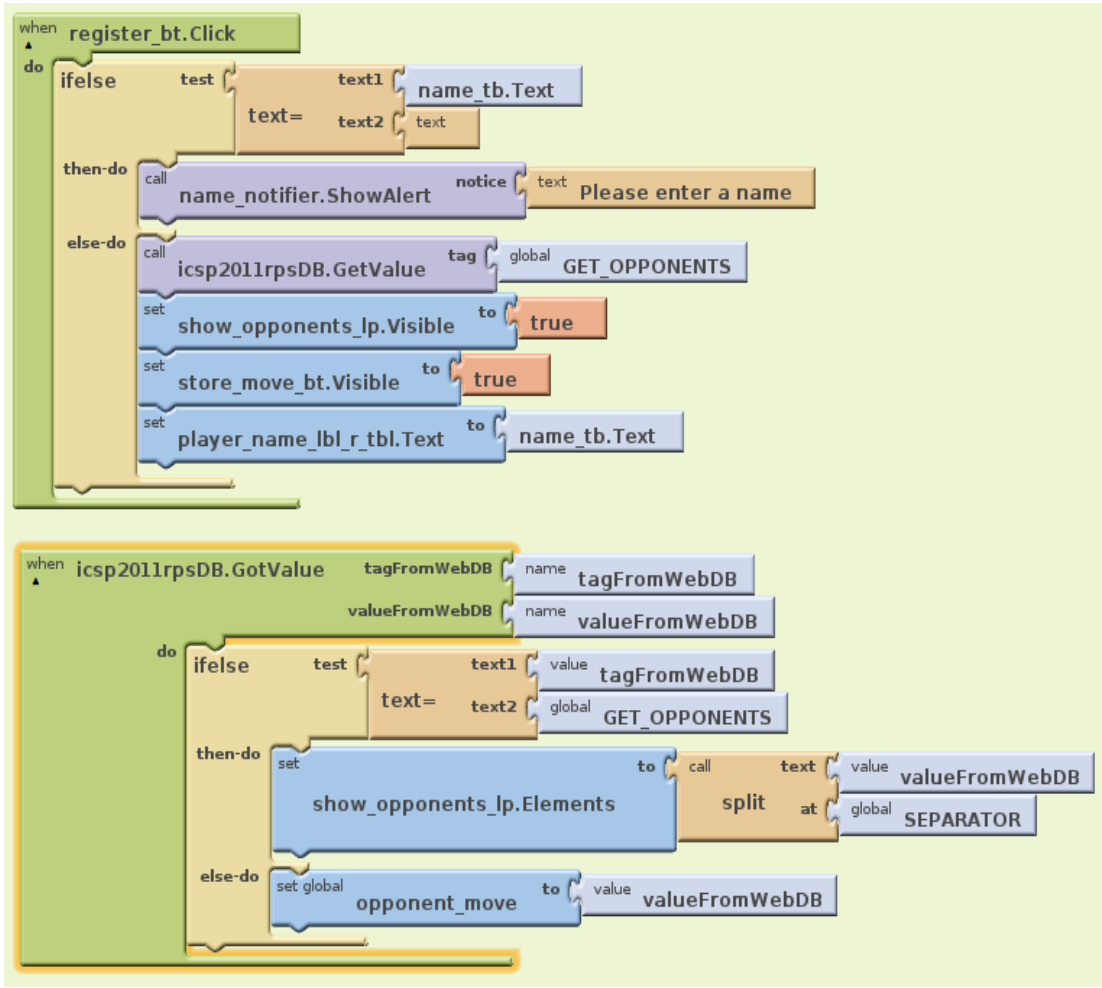
Initialize

- Define variables
- Assign values to variables
- Create the initial screen



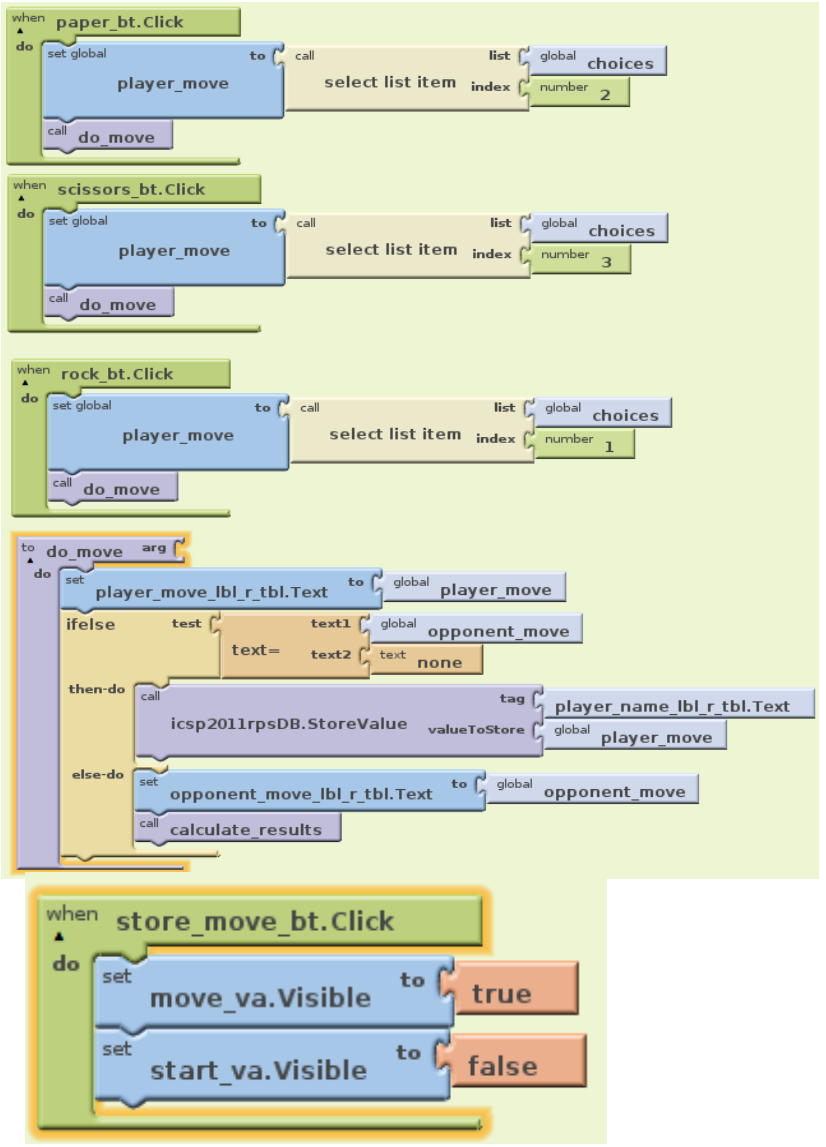
Create Register Functionality

- Register button click
- Database communication when getting a value
- Notification



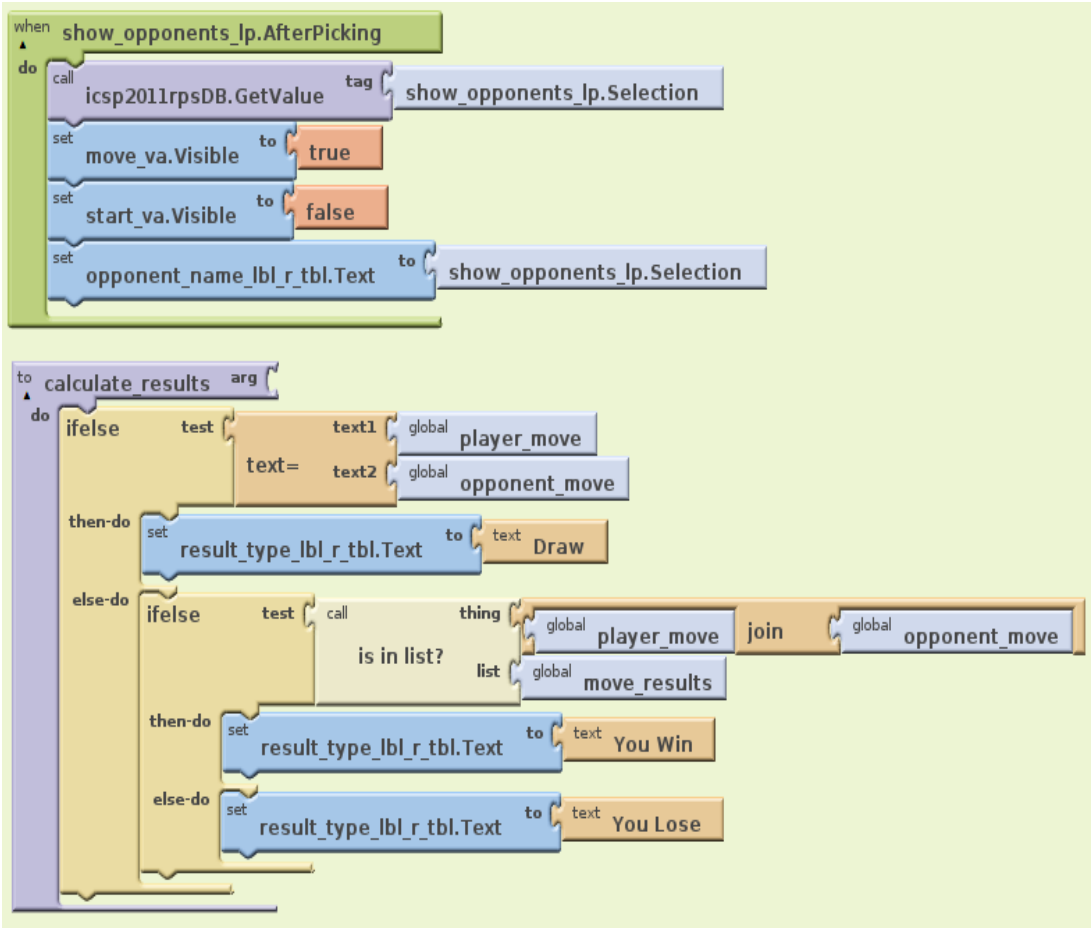
Store a Move

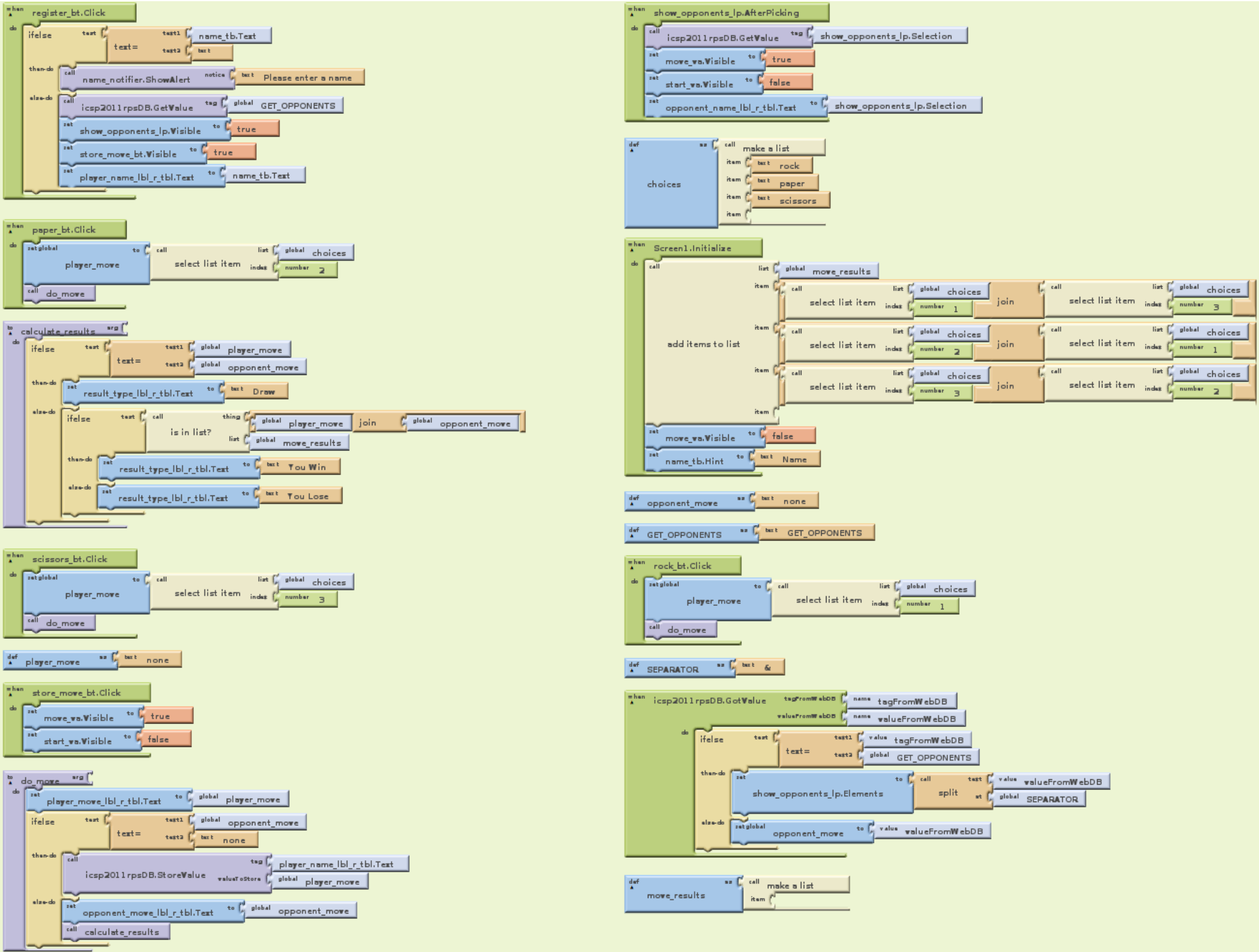
- Button choice behaviour
- Storing move in data base



Chose Opponents

- Methods after picking
- Calculating the result




when register_bt.Click
do
ifelse
text=
name_tb.Text
Please enter a name
name_notifier.ShowAlert
icsp2011rpsDB.GetValue
GET_OPPONENTS
show_opponents_lp.Visible
true
store_move_bt.Visible
player_name_lbl_r_tbl.Text
when paper_bt.Click
set global player_move
select list item
choices
number 2
do_move
to calculate_results
result_type_lbl_r_tbl.Text
Draw
is in list?
player_move
join
opponent_move
move_results
You Win
You Lose
when scissors_bt.Click
number 3
def player_move
none
when store_move_bt.Click
start_vs.Visible
false
to do_move
player_move_lbl_r_tbl.Text
opponent_move_lbl_r_tbl.Text
icsp2011rpsDB.StoreValue
valueToStore
when show_opponents_lp.AfterPicking
icsp2011rpsDB.GetValue
show_opponents_lp.Selection
move_vs.Visible
start_vs.Visible
opponent_name_lbl_r_tbl.Text
def choices
make a list
rock
paper
scissors
when Screen1.Initialize
add items to list
join
name_tb.Hint
Name
def opponent_move
def GET_OPPONENTS
when rock_bt.Click
number 1
def SEPARATOR
&
when icsp2011rpsDB.GotValue
tagFromWebDB
valueFromWebDB
show_opponents_lp.Elements
split
at
opponent_move
def move_results

Improvements of Rock, Papers or Scissors

- Disallow duplicate names
- Improve security e.g. Database injection
- Handle invalid values
- Return player to start when a move is stored
- Remove moves which has been in a game

AppInventor Notes

AppInventor Links

- http://appinventor.googlelabs.com/learn/tutorials/index.html
- http://appinventor.googlelabs.com/learn/reference/index.html
- https://sites.google.com/site/theairepository/
- https://sites.google.com/site/appinventorresources/home/tutorial-topics

Sample Apps

- General user base: https://sites.google.com/site/theairepository/source-code
- University of San Francisco: http://sites.google.com/site/usfandroidmarket/

- Debugging in AppEngine use
  - “do it”
  - “watch”
- AppEngine API not completly updated
- Do not use “,” when returning values with TinyWebDB
- Developing on a local server the address is “http://10.0.2.2”