Chromecast based Checkers Game, controlled by Android Device

Omid Omidi, Zach Almon, Matt Dunbar

University of Kentucky

Spring 2016

UK
UNIVERSITY OF
KENTUCKY
see blue.

Checkers on the Google Chromecast

- The Google Chromecast is a digital media device that turns a budget TV into an internet connected smart TV.

- Very Device Friendly to connect and stream content from devices to the TV.

- Very developer friendly with open source API and tool kits readily available for developers to use and create apps.

chrome
cast

The Goals

- To get familiar with Google’s different API’s
- To use Google’s open source code in conjunction with our own
- To create a game that can be used by many different people
- Checkers is a fun simple game that will be able to be implemented very easily
- The hard part is creating the code to communicate between the Hosting Server, The Chromecast, and the Android devices.

Requirements

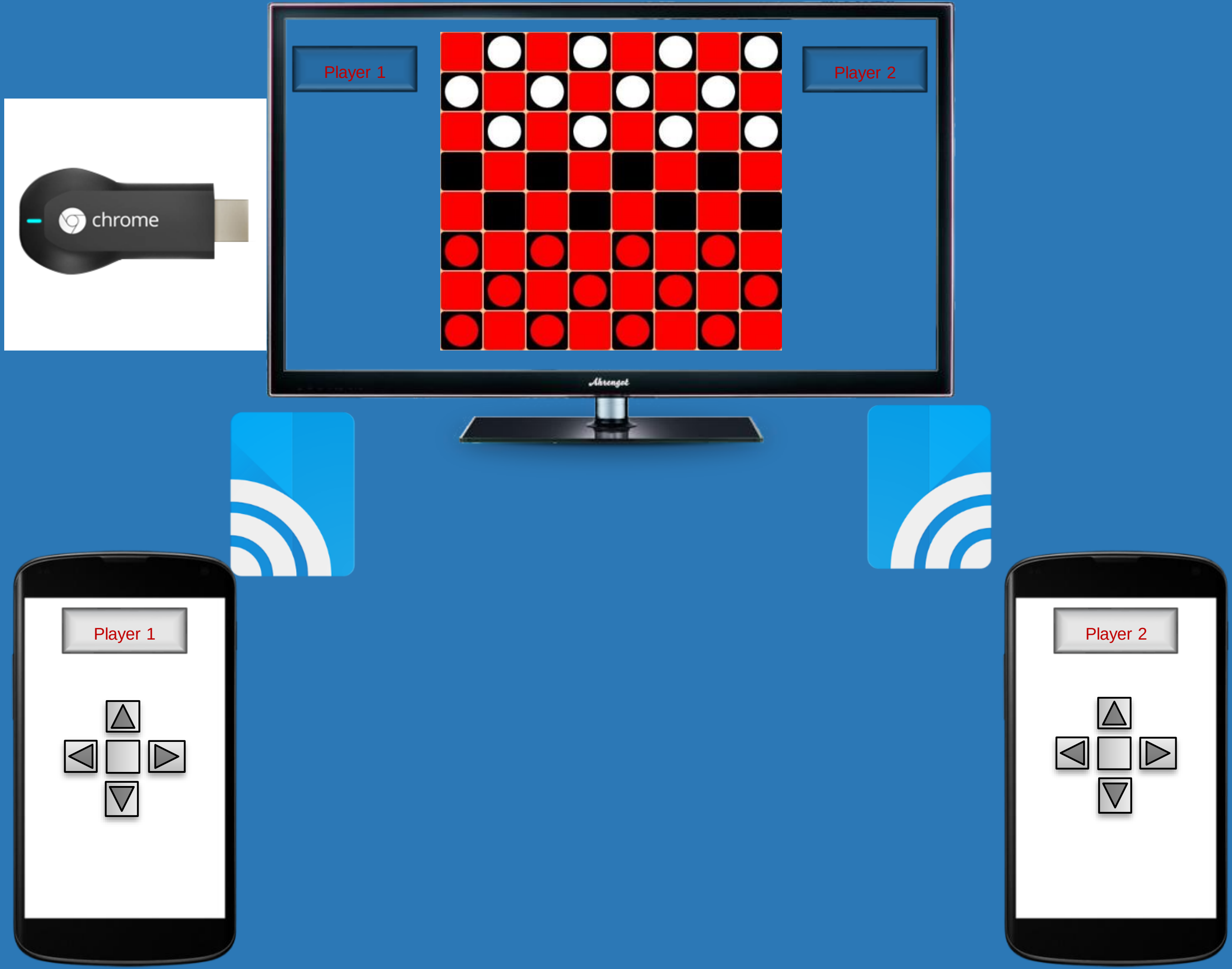
Android Controller App Specifications:

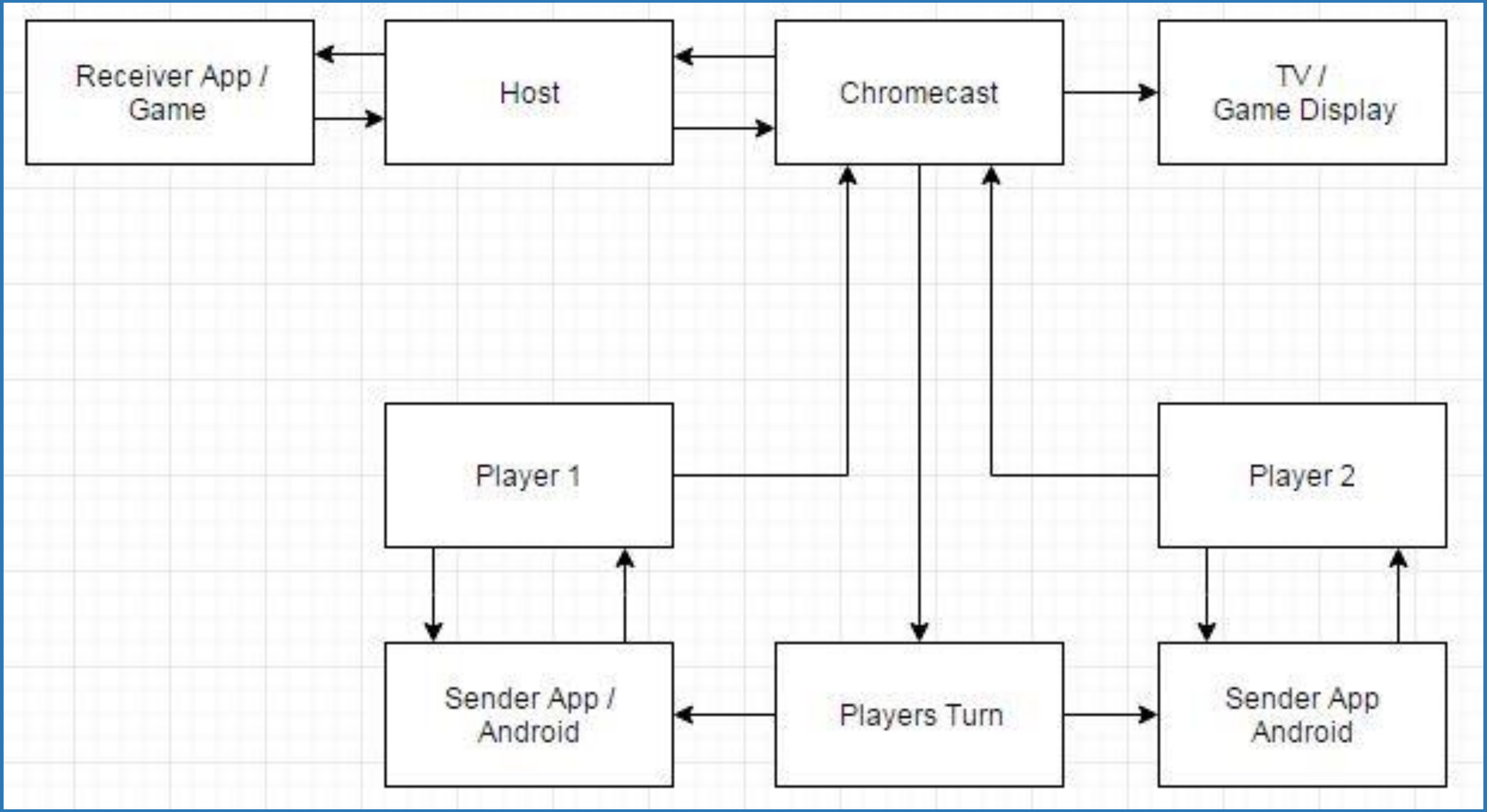
- Must have center select button
    - Must be able to select and deselect
- Must have four directional arrows surrounding the center select button
    - Must be able to go to all places, even in a multiple jump move
- Must have a back button to go back a screen to be able to switch which checker is selected
- Must be color coded to which player is red or black
- Must show when controller is disconnected from the Chromecast and show a reconnecting icon
- Must show what turn it is and whose turn it is on top
- Must be greyed out while other player is deciding their move

Requirements

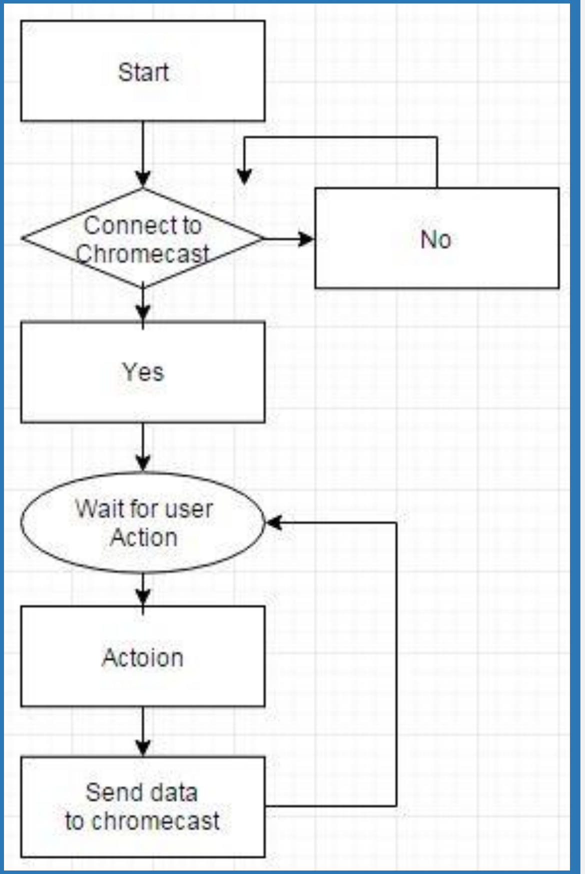
Chromecast Specifications:

- **Must have the checker board center screen big enough for all to see**
- **Must have player graveyards on either side of the board**
- **Must show what turn it is and whose turn it is on top**
- **Must be color coded to match controllers to which player is red or black**
- **Must show when a controller is disconnected from the Chromecast and show a reconnecting iron**
- **Must show all possible moves available, including placement after jumps**
- **Must show selected checker as a highlighted piece**

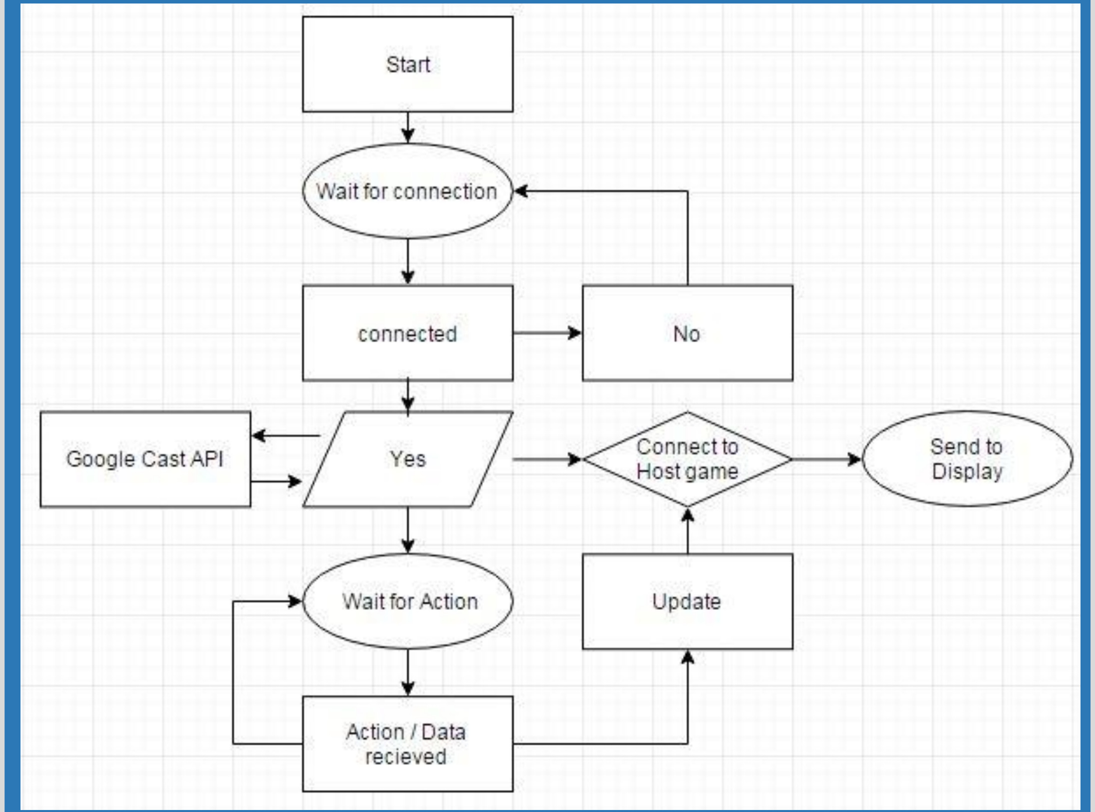

chrome
Player 1
Player 2
Player 1
Player 2


Receiver App / Game
Host
Chromecast
TV / Game Display
Player 1
Player 2
Sender App / Android
Players Turn
Sender App Android

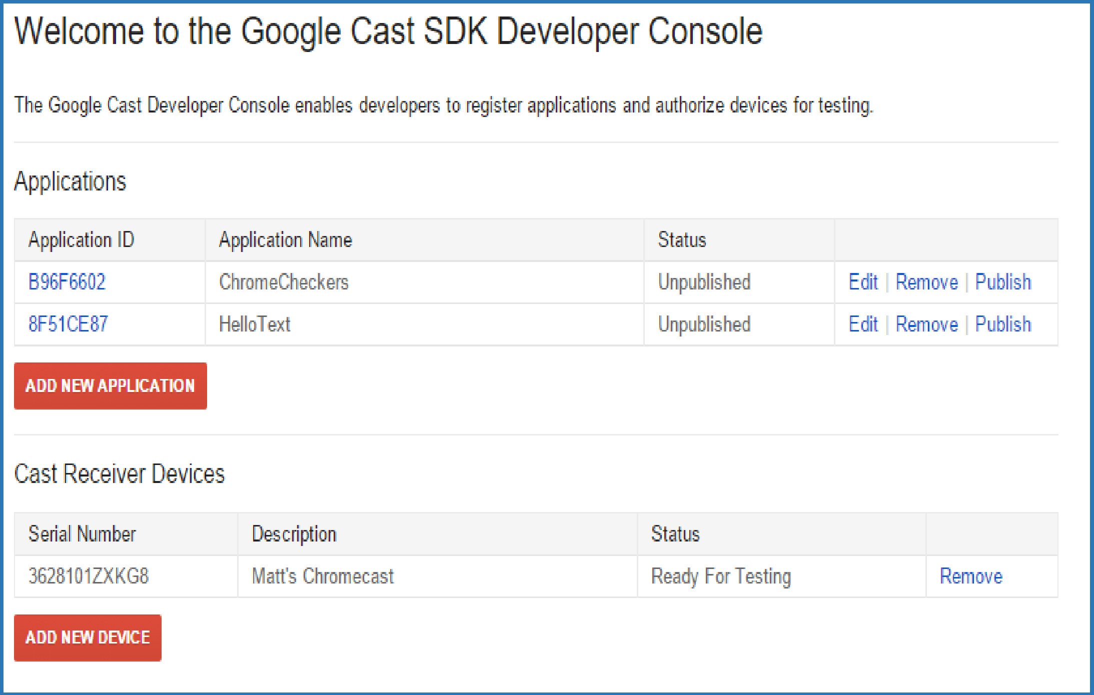
Sender App



Reciever App



Google Cast SDK Developer Console



Welcome to the Google Cast SDK Developer Console

The Google Cast Developer Console enables developers to register applications and authorize devices for testing.

Applications

Application ID	Application Name	Status	
B96F6602	ChromeCheckers	Unpublished	Edit | Remove | Publish
8F51CE87	HelloText	Unpublished	Edit | Remove | Publish

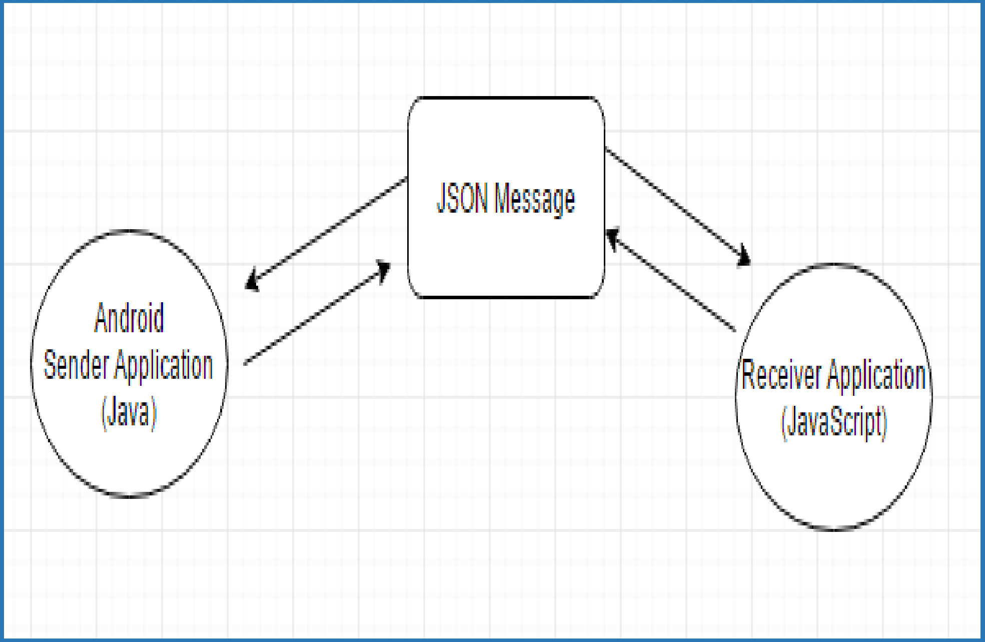
Cast Receiver Devices

Serial Number	Description	Status	
3628101ZXKG8	Matt's Chromecast	Ready For Testing	Remove

- To make the Chromecast run your application at all, you have to register your Chromecast serial number to the developer console.
- Also, for every application that you make, there has to be an application ID associated with it that also has to be hard coded into both the receiver and sender applications.
- It costs $5 for a Chromecast Developer license.

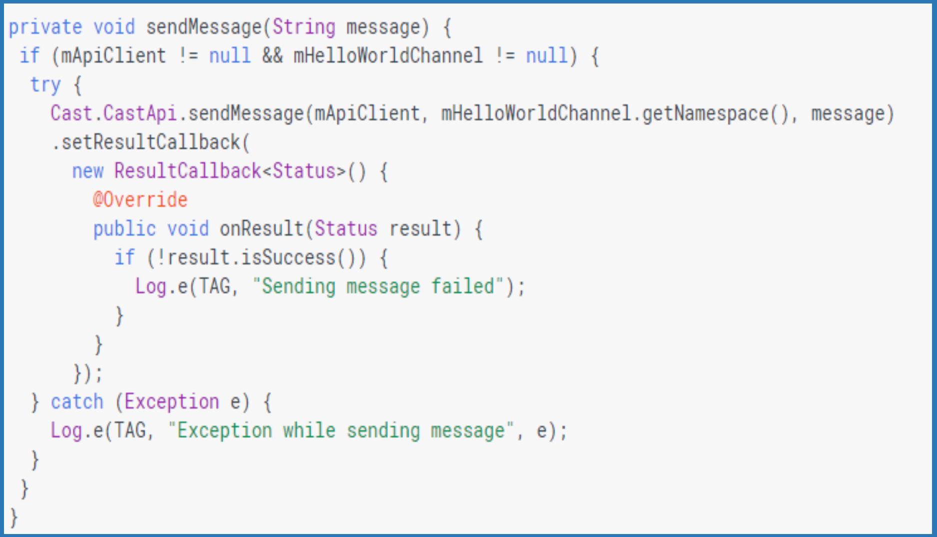
Java + JSON + JavaScript

- The Chromecast API, in the way that we're using it, basically combines these 3 together.
- Java on the Android side handles the JSON messages being sent from the sender application (the Android app) and hands it over to the JavaScript in the receiver application.


JSON Message
Android Sender Application (Java)
Receiver Application (JavaScript)

CastReceiverManager & Sender App Integration

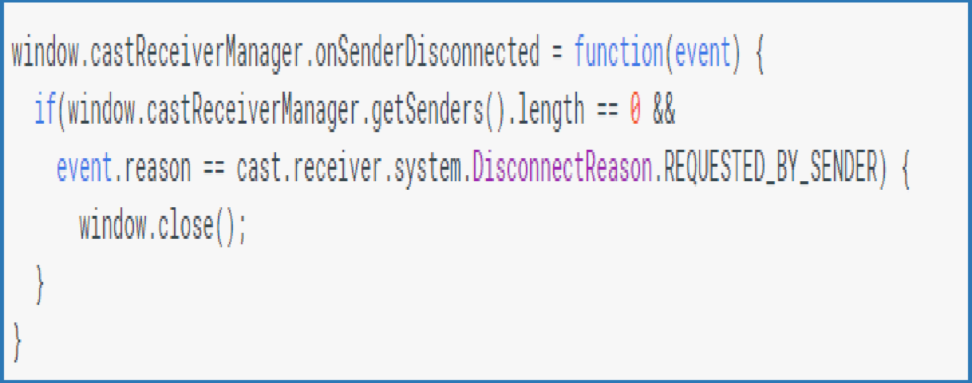
- CastReceiverManager: does most of the work with handling the sending and receiving of messages when it comes to the receiver application.
- After this is initialized, the sender app creates a communication channel "Cast.CastApi.setMessageReceivedCallbacks" and will be able to receive feedback from the ReceiverManager every time it sends a message with "Cast.CastApi.sendMessage"



```
private void sendMessage(String message) {
 if (mApiClient != null && mHelloWorldChannel != null) {
  try {
    Cast.CastApi.sendMessage(mApiClient, mHelloWorldChannel.getNamespace(), message)
    .setResultCallback(
      new ResultCallback<Status>() {
        @Override
        public void onResult(Status result) {
          if (!result.isSuccess()) {
            Log.e(TAG, "Sending message failed");
          }
        }
      });
  } catch (Exception e) {
    Log.e(TAG, "Exception while sending message", e);
  }
 }
}
```

Session Management

- An important part of both applications, if not the most important part is the session and session management features.
- The Player 1 sender application creates the session, and from there the session is running on the receiver application, once this happens Player 2 sender can connect.
- If for any reason Player 1 or Player 2 get disconnected from the session, there is code to handle this. So in this case we would pause the game, keep whoever’s turn it is stored and set a timeout. If the player does not reconnect within 2-3 minutes, they forfeit the game to the other player.



```
window.castReceiverManager.onSenderDisconnected = function(event) {
  if(window.castReceiverManager.getSenders().length == 0 &&
    event.reason == cast.receiver.system.DisconnectReason.REQUESTED_BY_SENDER) {
      window.close();
  }
}
```

Conclusion

- Hopeful Extras for this Project:
  - Support for Apple products
  - Single player support (Play against the computer)
- To create a game that can be used by many different people
- Checkers is a fun simple game that will be able to be implemented very easily
- Creating the code to communicate between the Hosting Server, The Chromecast, and the Android devices.

?

An Equal Opportunity University

UK
UNIVERSITY OF
KENTUCKY®
see blue.