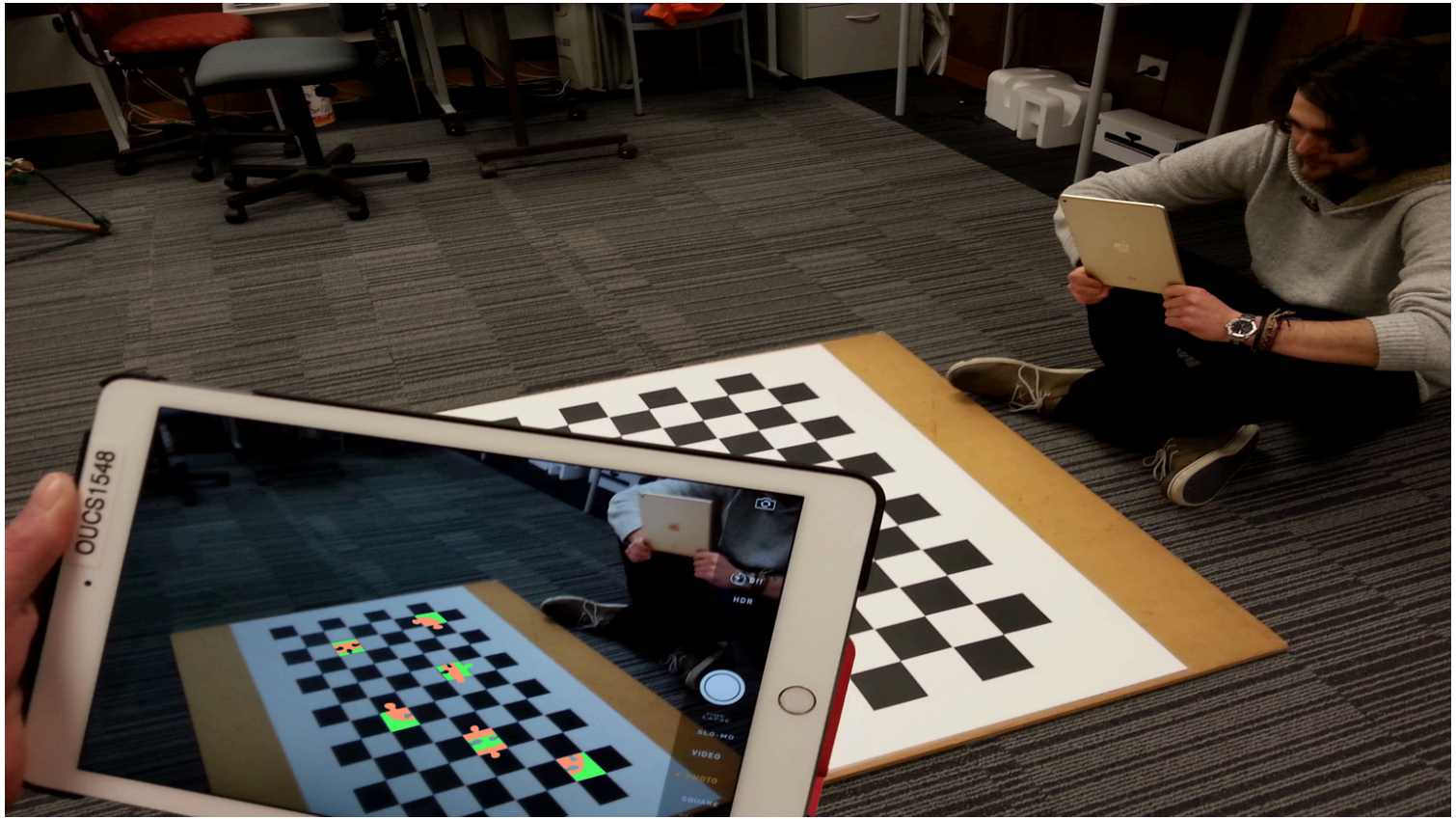
Giguesaur

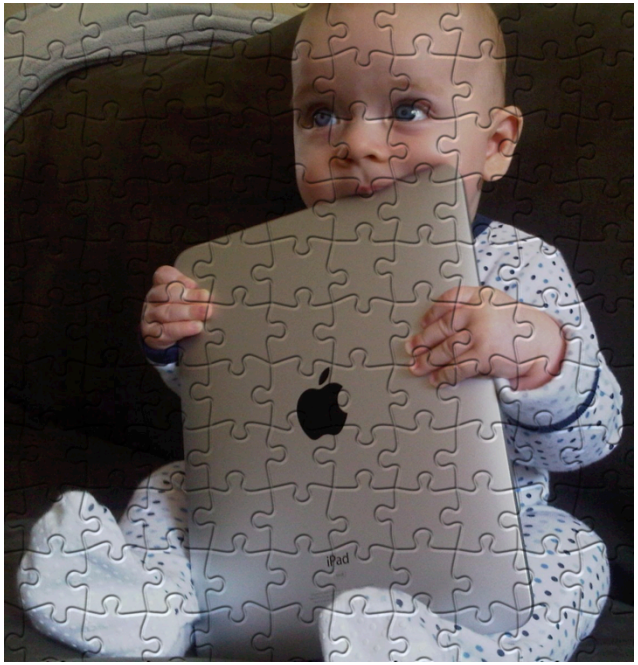
Josh, Ash and Shahne
Supervisors: Geoff Wyvill and David Eyers

Introduction



Introduction

- Localisation - Josh
- Game Logic - Ash
- Networking - Shahne



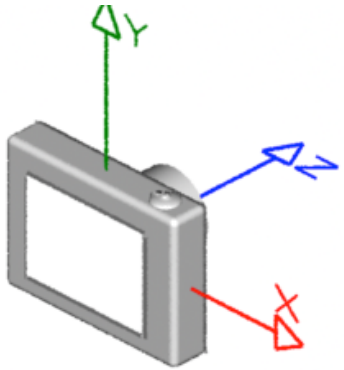
Giguesaur iPad Localisation

Joshua La Pine

What is Localisation?

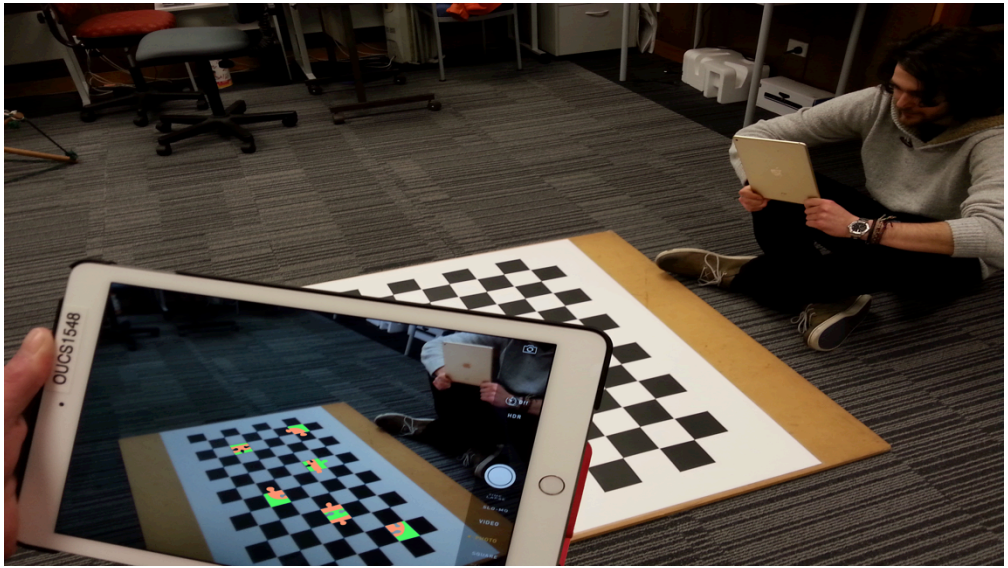
- Locating the camera’s position relative to some real world object. In this case the puzzle board.

- The camera’s location is defined by its translation and rotation in x,y,z relative to the game board.



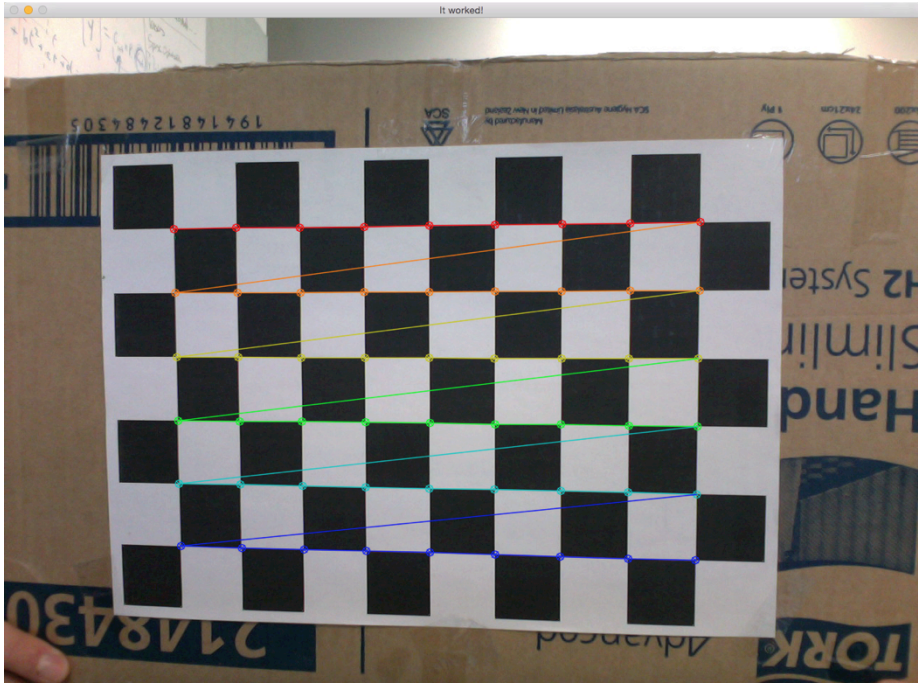
Why Localisation?

- In order to render the puzzle pieces superimposed correctly on the board the camera’s position must be known.



How is it done?

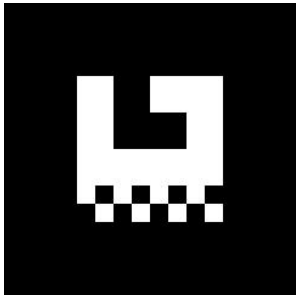
- Pose Estimation
- Using a calibrated camera
- Locate points in your image that correspond to known points in world space.
- Run SolvePnP



Marker Tracking

- Researched different marker tracking techniques: ARToolkit, QR Codes, etc.

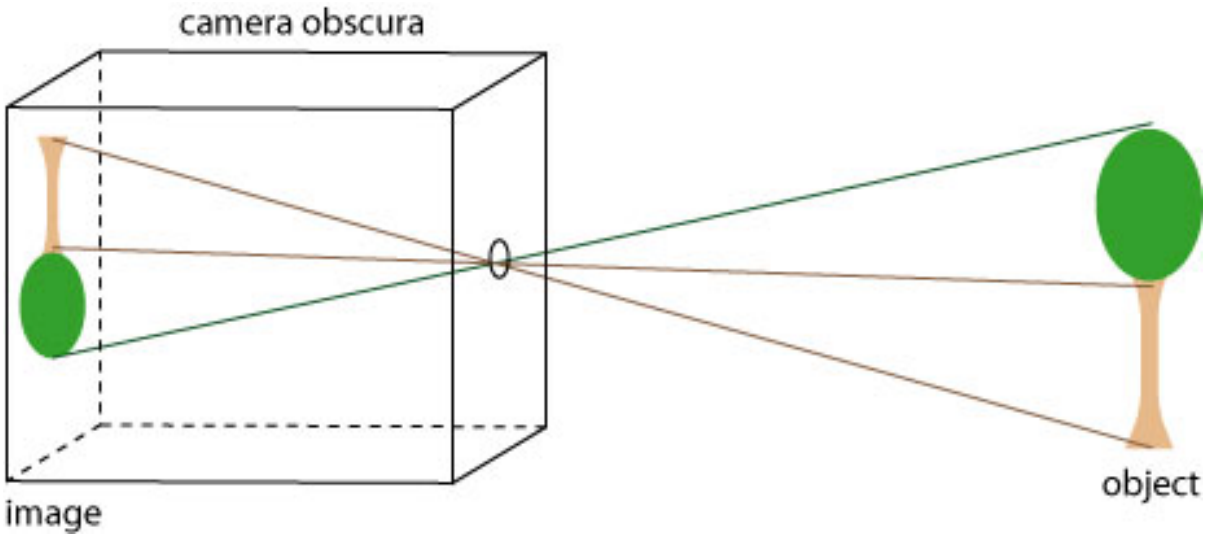
- Image processing is required, along with camera calibration.



OpenCV

- Open Computer Vision Library.
- Image I/O, processing, camera access, pose estimation, camera calibration.
- iOS Support.

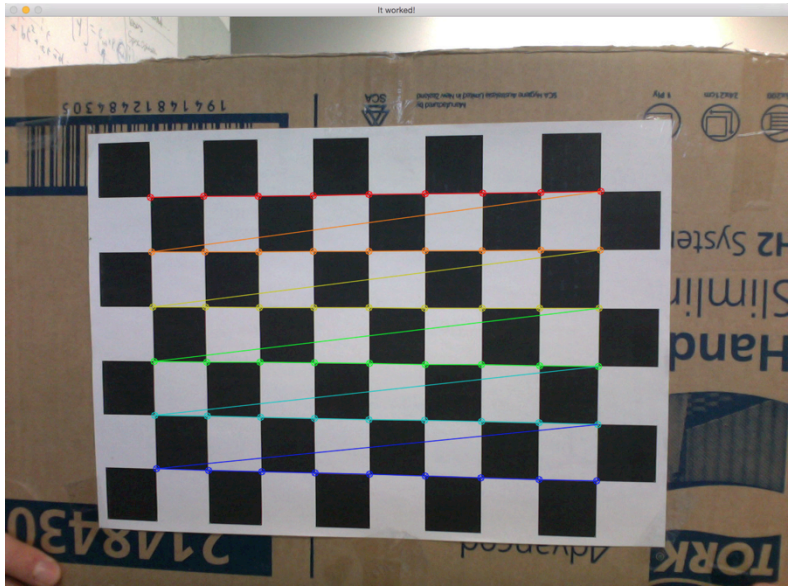
Camera Calibration



Camera Calibration

- Why? Accounts for several kinds of distortion. This causes problems for any kind of image processing.

- How? Using a predefined calibration pattern.
- Large number of points that can be identified.



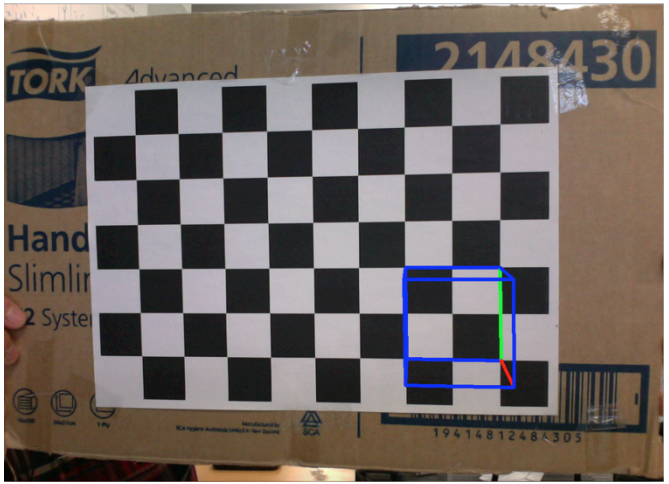
Camera Calibration

- Locates internal corners of the chessboard image across a number of calibration images. I use between 15-20.

- Calculates 2 matrices, one with the camera matrix and the other for the distortion coefficients. Loaded at the start of the program.

Augmented Reality

- Ideal: Set up OpenGL Camera
- Currently: Find internal corners of chessboard, calculate rotation and translation vectors.
- Use to transform model vertices defined in world space coordinates into image space coordinates and render a cube on the board.



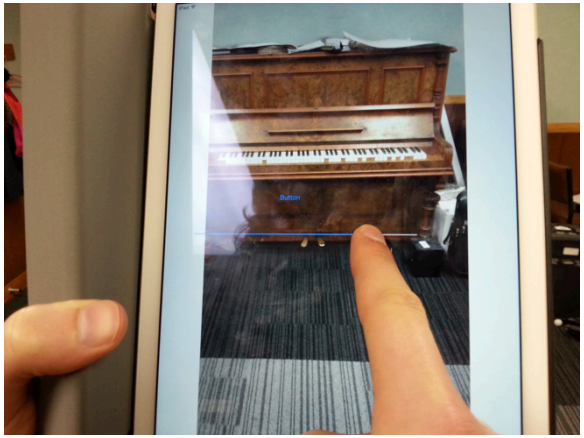
iOS

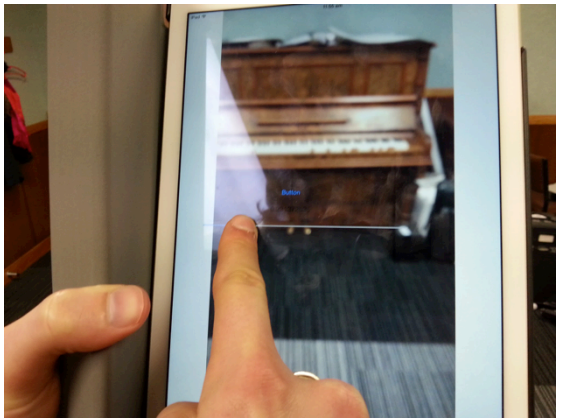
- Operating system for Apple’s mobile devices, such as the iPhone, iPad, and iPod Touch.

- Different Environment from iMac, rules are a little different. Resulting in different API calls and methods.

iOS

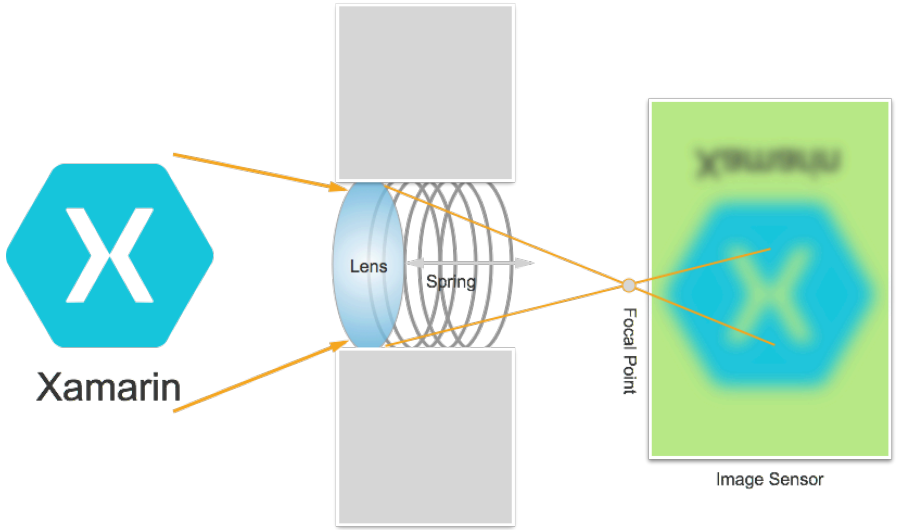
- Started with CVVideoCamera.
- Discovered variable focus camera
- Set up native iOS camera routines
- Began experimenting with different focus levels





iOS

- Still an open problem.
- Setting value might not be a consistent operation.
- Current idea is to generate a set of calibration parameters for different focal values and to set the focal value to the nearest value with stored parameters.



Limitations and Future Work

- Solve the calibration problem.
- Whole board must be in view, different tracking markers.
- AR that can handle partially obstructed tracking markers.
- Open it up to a floor based game so a larger number of people can play.

Giguesaur Game Logic

Ashley Manson

My Part

- Game Logic: What the puzzle is
- Rendering the puzzle using OpenGL: How the puzzle is shown on screen
- User Interface: How the user interacts with the game

Background - Jigsaw Puzzle Games

- Jigsaw Puzzle Games have been done
- Most have an orthographic projection
  - Looking top down at the puzzle, a little boring
- Are limited to where puzzle pieces can go
  - To solve them, the pieces need to be put in a grid

Background - Jigsaw Puzzle Games

Source: http://ios.appeero.com/a/farm-animals-puzzles-jigsaw.html

Source: http://www.bigfishgames.com/games/65/elitejigsawpuzzle/

Background - Augmented Reality

- The idea of superimposing game objects in the real world

- Some games have done this
  - PulzAR (PS Vita)
  - ARSoccer (iOS)
  - ARDefender (iOS, Android)

Background - Augmented Reality

Source: http://www.videogamer.com/psvita/pulzar/screenshot-1.html

PulzAR

Source: https://www.youtube.com/watch?v=rB5xUStsUs4

ARDefender

Progress So Far

- Game Logic mostly done
- Rendering squares with textures
- Pieces being picked up and placed back down
- Pieces can be rotated
- Pieces snapping together
- Puzzle can be solved

Game Logic

- A piece has an ID, an x, y location, and rotation
- A piece has four edges
- An edge can be closed, open, or invalid
  - closed = joined its neighbour
  - open = not joined its neighbour
  - invalid = has no neighbour

Game Logic

- Jigsaw Puzzle with a number of Puzzle Pieces
- A board to place pieces on
- A player can hold one piece
  - Stops them from stealing all the pieces from other players

Interface

- The player taps a piece to pick it up
- The piece is put in the player’s inventory (shown in bottom left of the screen)
- Currently, the player taps the screen where they wish to place the piece
- In future, we want the player to position and rotate the iPad to place piece

Neighbouring Pieces

- Most jigsaw puzzle games have a grid to place pieces
- The Giguesaur puzzle pieces can be placed and solved anywhere on the board
- Each piece needs to join up to its neighbour rather than be placed on a grid

Rendering the Puzzle

- A piece’s parameters determine where on the board they appear

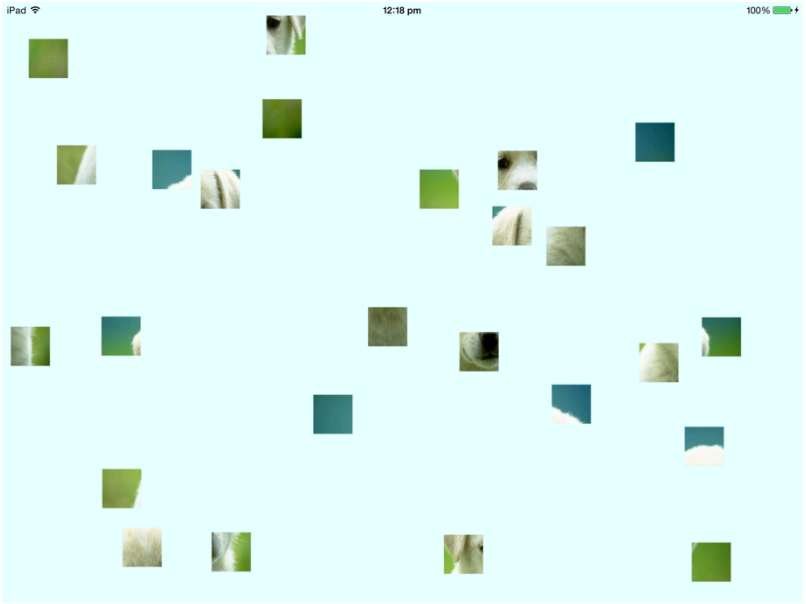
- Started with squares
  - Now they have textures

- Want to have more interesting polygons with curved edges in future

Snapping Pieces

- Pieces can move around the board
- They need to join up to each other
  - All pieces joined together means puzzle has been solved
- Two pieces that are close enough to each other will snap together
  - A piece can only snap to its neighbour piece

Giguesaur Screenshots

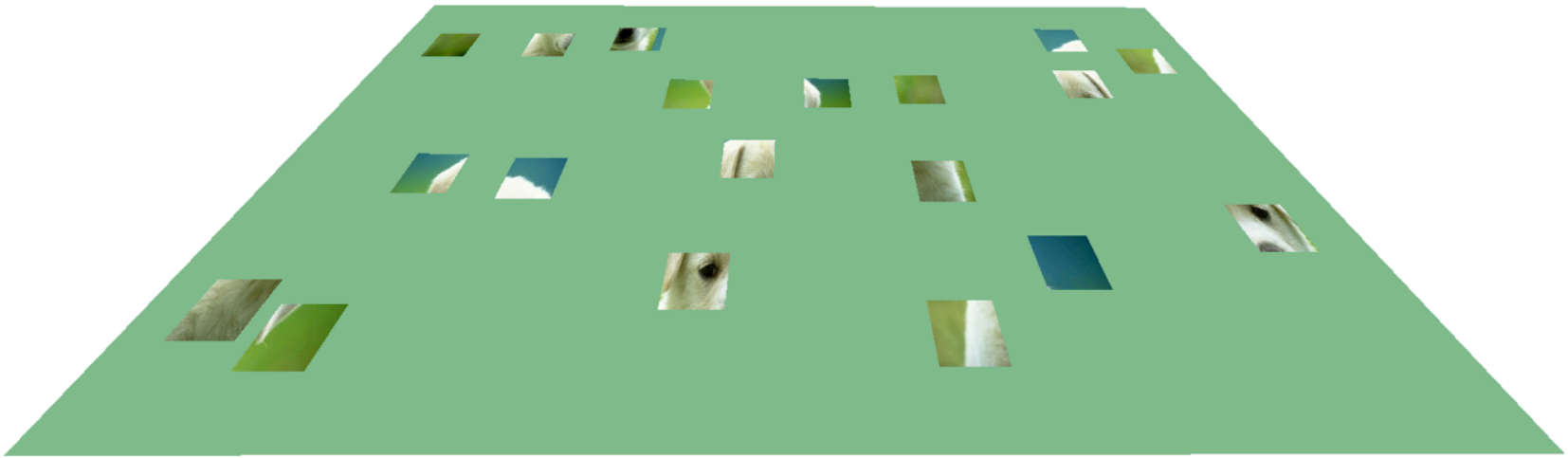


Puzzle Layout when first started

Puzzle being solved

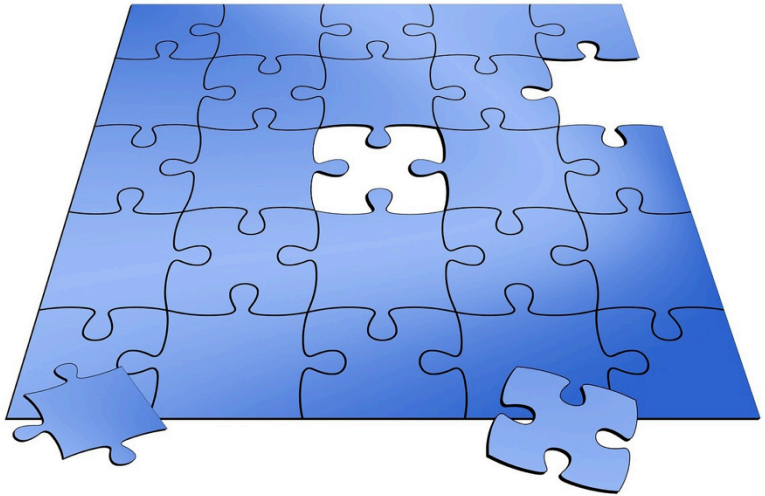
Giguesaur Screenshots

Giguesaur - Perspective Screenshot



Things to do

- Curved edges to pieces, making them look more like puzzle pieces
- Have the player position the iPad to place a piece



Source: https://pixabay.com/en/jigsaw-puzzle-missing-piece-313585/

Things to do

- Integrate my part of the project with the others
  - Shahne’s part, allowing more than one user to play a single instance of a game
  - Josh’s part, I’ll create the perspective view from the camera position he gets

- Bug fixes and testing

Giguesaur Networking

Shahne Rodgers

Problem

- All iPads need to have a consistent ‘game state.’

- What if two people try to pick up the same piece?

Communication Protocol

- Bluetooth
- WiFi
- Others?

Client-Server Model

- Server holds the board state (image, number of pieces and their locations).
- This is replicated to the clients.
- Clients only contact the server when they pick up or drop a piece.

Connecting to the Server

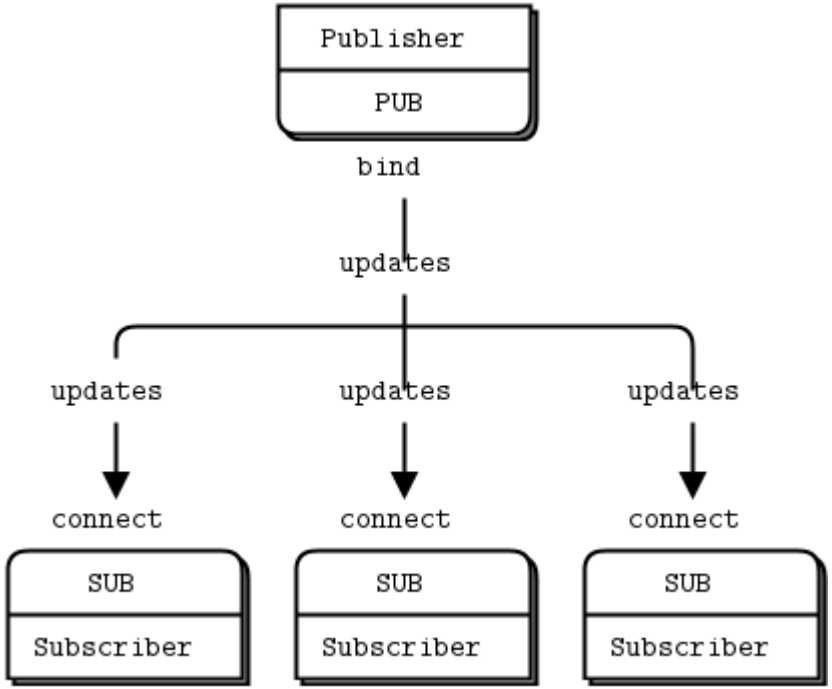
- IP Address: 192.168.67.1 or fe80::e9cb:894c:ca3e:8967%25

- Bonjour

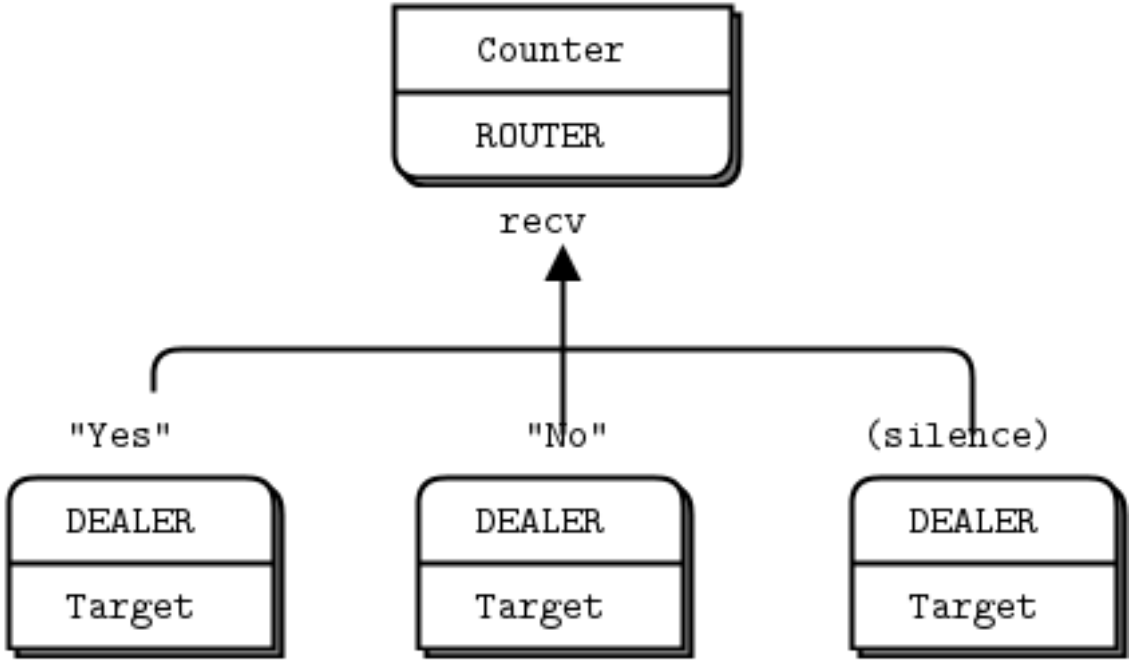
Communication APIs

- NSStream and GameKit (Apple)
- ZeroMQ

Communication Patterns - Pub/Sub

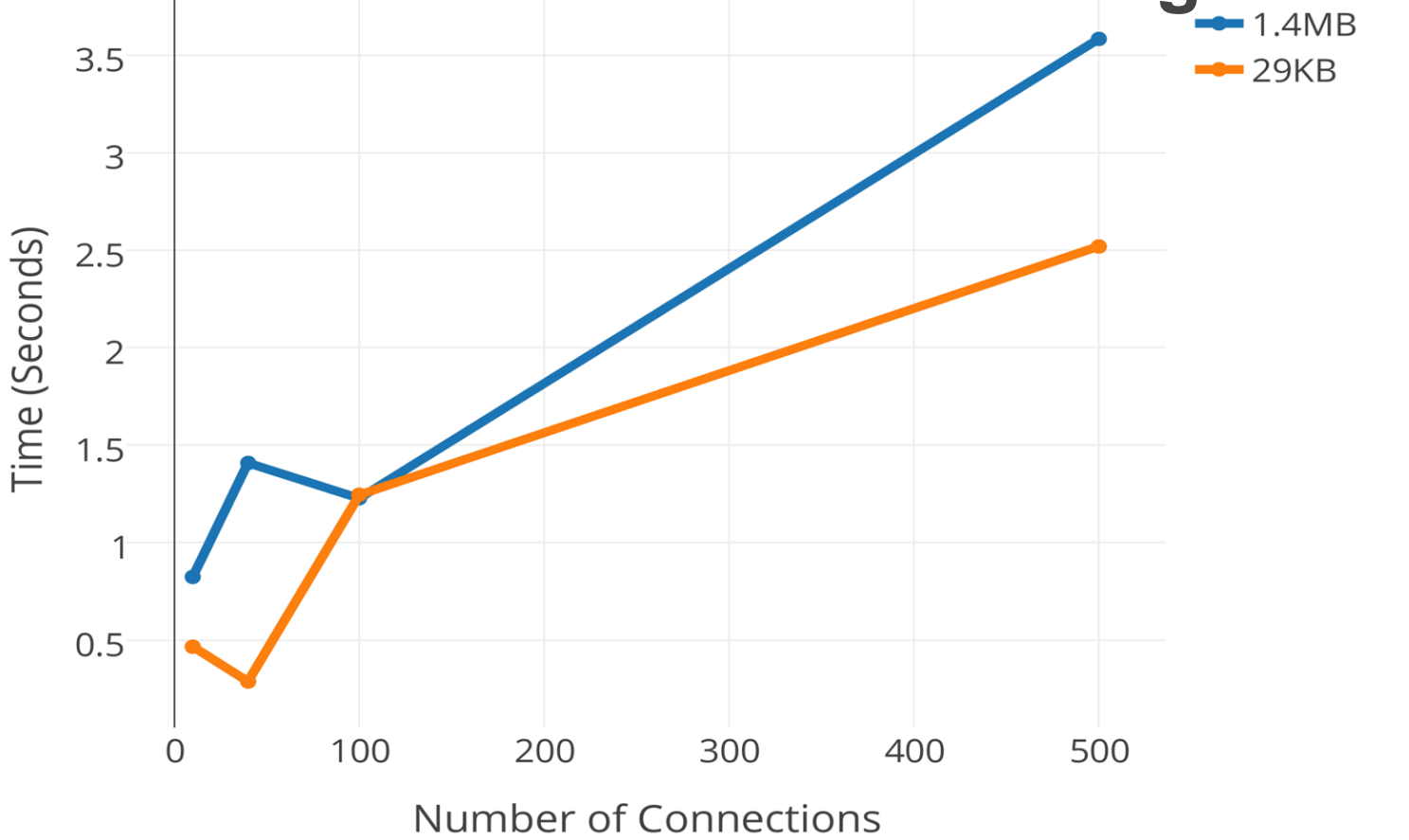


Communication Patterns - Router/Dealer

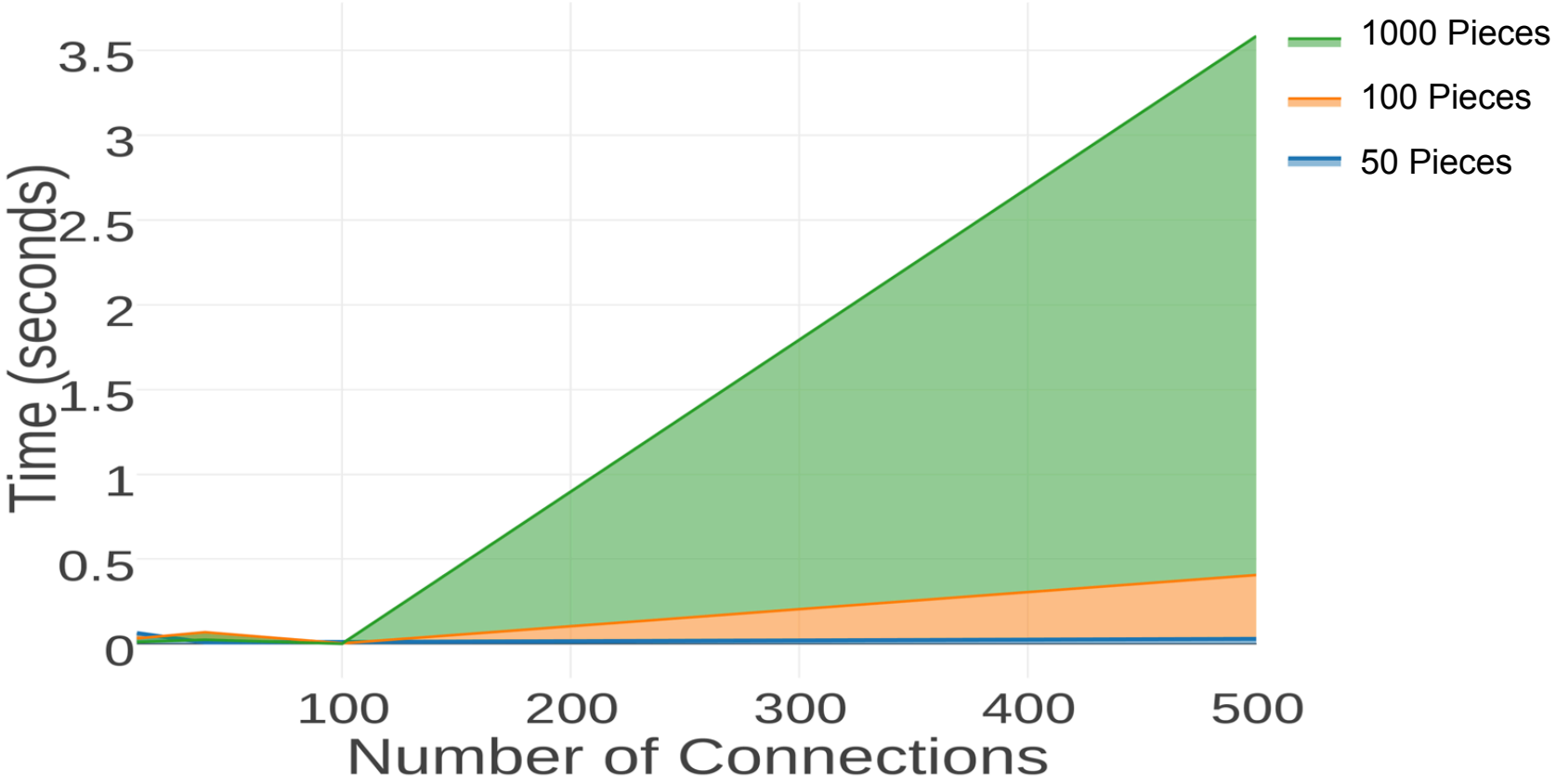


The Missing Piece

Time Receiving Board With Varying Numbers of Connections and Image Size



Time to Receive Response Based on Number of Pieces and Number of Connections



Future Work

- Integrate networking code with Josh and Ash’s code.
- More testing.
- Improve speed of receiving initial board state.
- Minor fixes: client naming, piece dropping, server’s user interface.
  - Consensus Networks.