# QuadrupleScore

Jose Lemus, Billy Witrock, Saad Mazhar, Irving Pena

# Introduction

- The purpose of our project was to create a variation on the classic game Scrabble
- For those who don't know, here are the rules of Scrabble:
  - 15 x 15 master board
  - Each player is distributed 7 tiles from a bag of 100 tiles initially. This bag replenishes players as they use up their tiles
  - Players take turns trying to make a word with the letters they are given. The word must use a letter already on the board
  - The game ends when either no tiles are left in the bag or when no one can make a move
- To make it more interesting
  - Now each player has their own version of the board and can make moves across a distributed system, thus removing the requirement of everyone being in the same room
  - In our version of Scrabble, players simultaneously race to put their tiles down first. There are no turns and there is no waiting
  - Other players' words appear on your screen as soon as they are deemed valid, and anyone who is in the middle of their turn when a word is placed down will have their tiles placed back in their hand

# Minimum and Maximum Deliverables

- Minimum Deliverable
  - Have a distributed game of Scrabble working using Python and Erlang
  - Players would interact with the game via the Linux terminal instead of a GUI, and the board and tiles would be represented in ASCII

- Maximum Deliverable
  - Have players interact with a GUI
  - Have the option of single-player Scrabble, where a player would play against an AI whose difficulty could be set. Depending on the difficulty, the AI would split the board into multiple threads, find all the possible words in a section of the board, and either play the move with the highest score (when AI set to difficult), or chose a move at random (when AI set to easy)

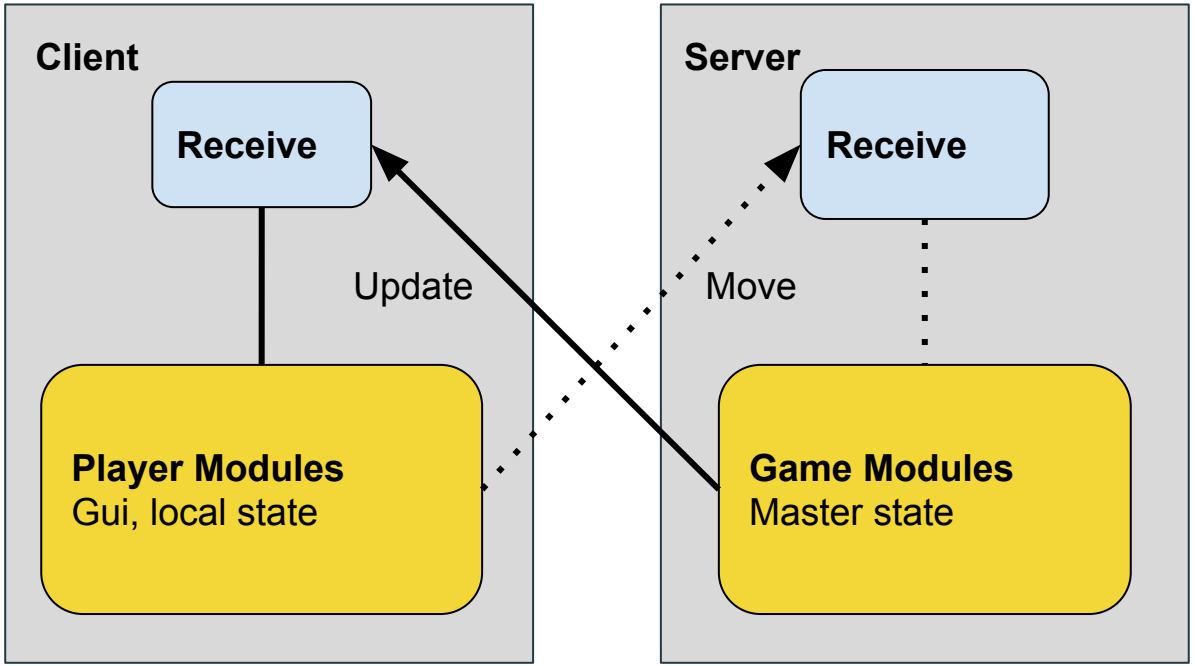
# Demo

# Major Design Choices

- **Using Erlang to only send messages. No game logic is handled**

- Middle modules

# Distribution



# Messages

| Move | |
|---|---|
| **Key word** | move |
| **Player PID** | player@lab118i |
| **Word** | hi |
| **Starting position** | (0,0) |
| **Direction** | right |
| **Used Tiles** | [(value, score, multiplier, id)...] |

| Update | |
|---|---|
| **Key word** | refresh |
| **Board** | [[‘0’, ‘t’ …]<br>[‘0’, ‘a’, …]<br>…] |
| **Scores** | [4, 10, 28, 8] |
| **Old tiles** | [(value,…)...] |
| **New Tiles** | [(value,...)...] |

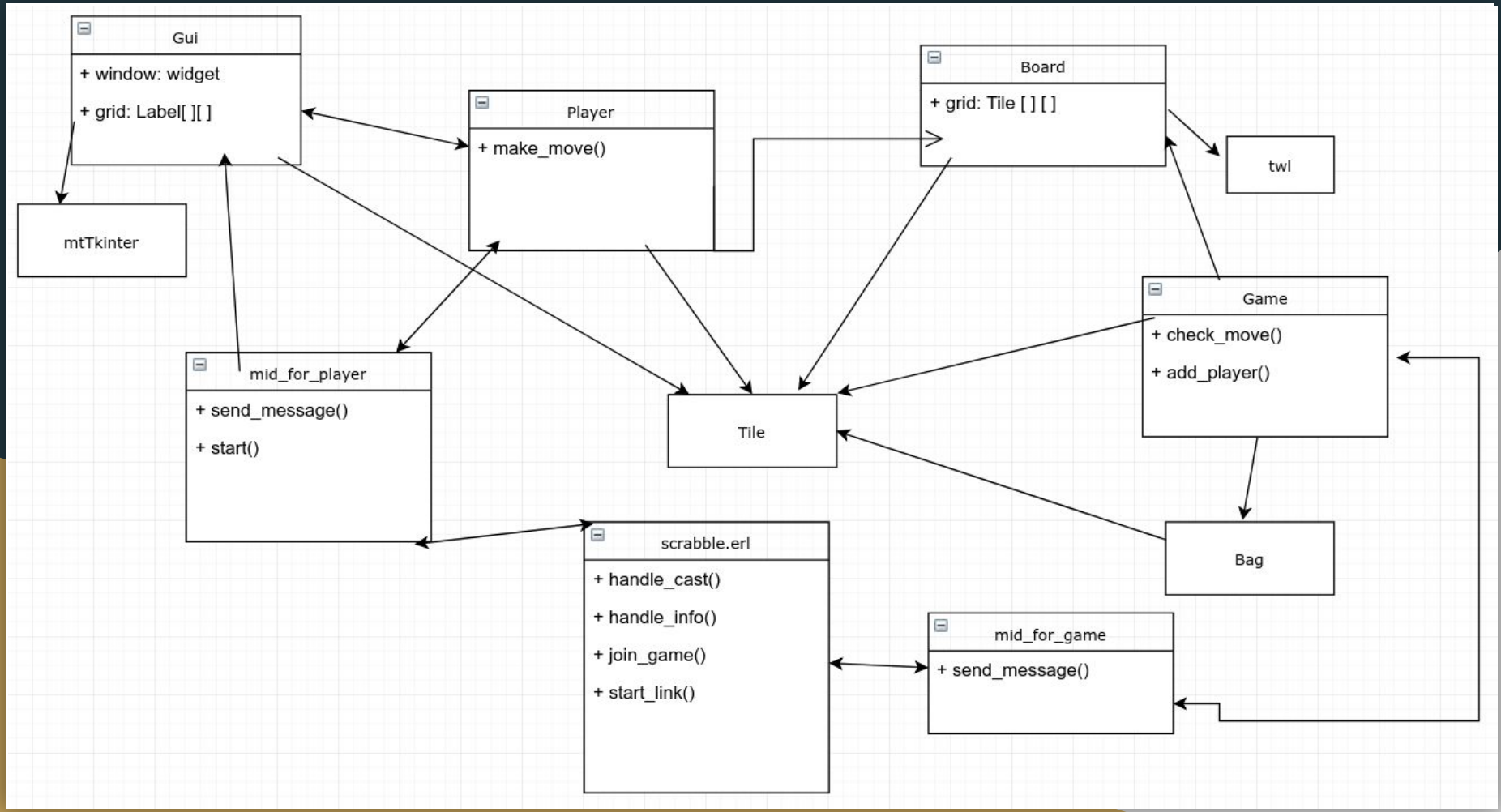
# How is everything connected?

- How do they interact?
- What modules are used?
- What classes are used?

# Major Design Choices

- Erlang only to send messages no game logic

- **Middle modules**

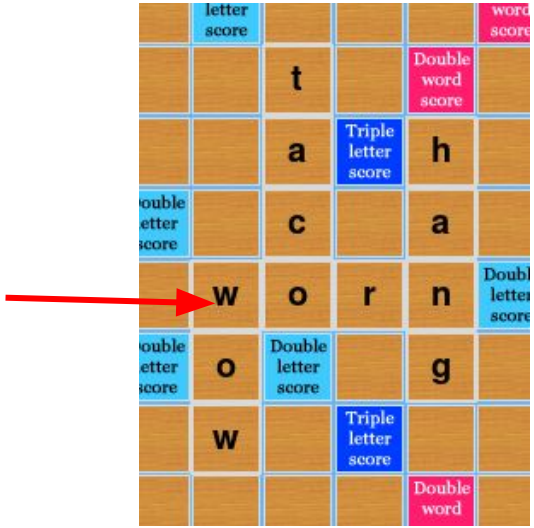

Gui
+ window: widget
+ grid: Label[ ][ ]
Player
+ make_move()
Board
+ grid: Tile [ ] [ ]
twl
mtTkinter
Game
+ check_move()
+ add_player()
mid_for_player
+ send_message()
+ start()
Tile
scrabble.erl
+ handle_cast()
+ handle_info()
+ join_game()
+ start_link()
Bag
mid_for_game
+ send_message()

# How a score is calculated

- What happens if hypothetically the word “worn” has just been placed? (yes it is impossible, but let's just play along)

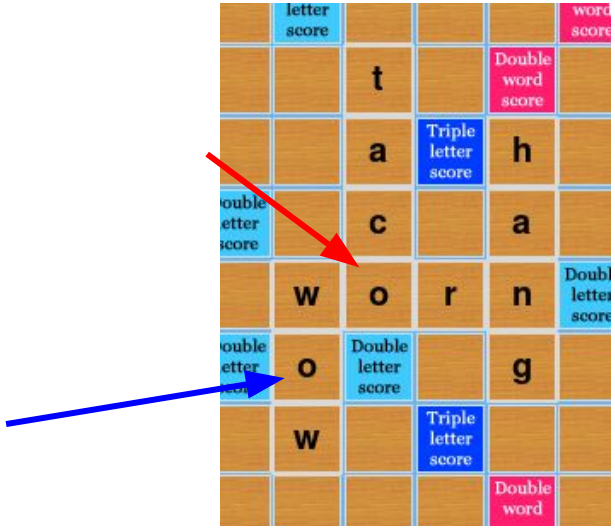
# How a score is calculated

- What happens if hypothetically the word “worn” has just been placed? (yes it is impossible, but let's just play along)



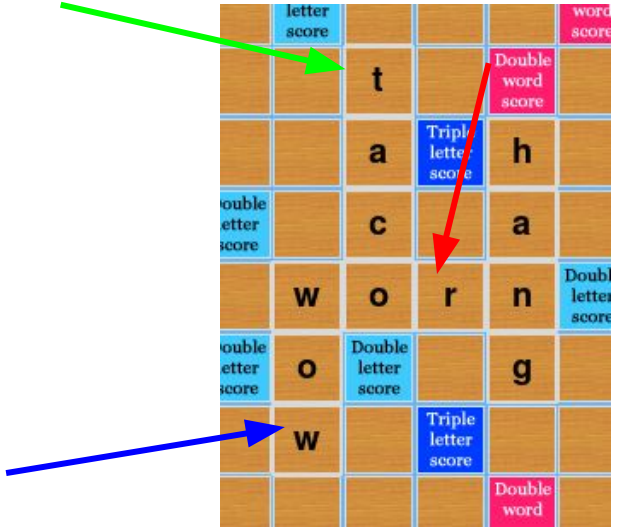
# How a score is calculated

- What happens if hypothetically the word “worn” has just been placed? (yes it is impossible, but let's just play along)



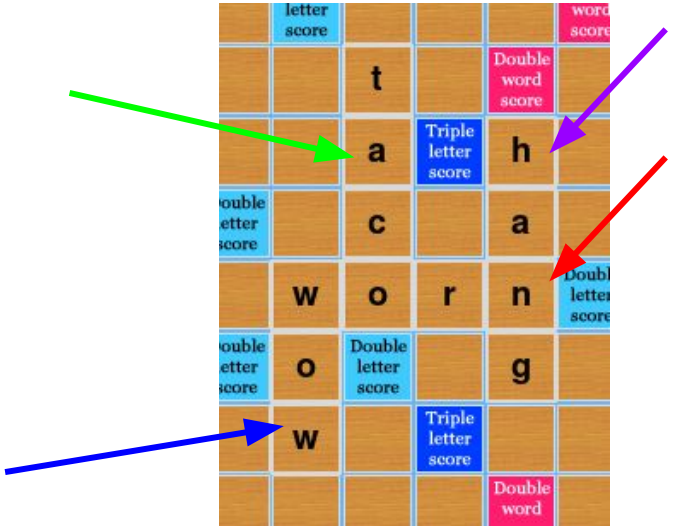
# How a score is calculated

- What happens if hypothetically the word “worn” has just been placed? (yes it is impossible, but let's just play along)



# How a score is calculated

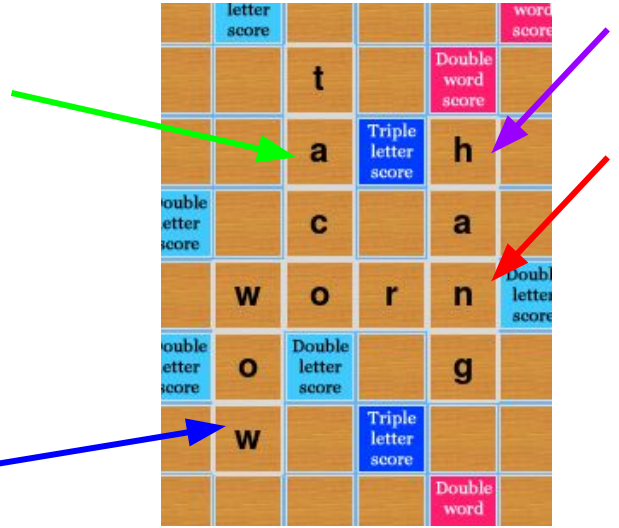
- What happens if hypothetically the word “worn” has just been placed? (yes it is impossible, but let's just play along)



# How a score is calculated

- What happens if hypothetically the word “worn” has just been placed? (yes it is impossible, but let's just play along)

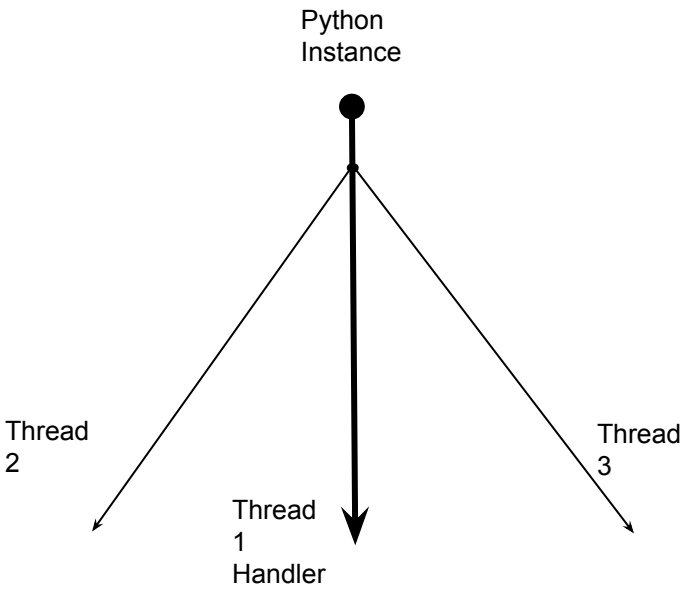
Score = 12
Main multiplier = x1
Extra score = 0
Is_valid = True
Mutex = Lock()



# Large Roadblocks We Encountered

# Erlport Problems

- Erlport
  - Installing
    - Use code from pull request
  - Some caveats
    - Handlers
      - Main thread
      - Fine for server, not for player
    - Calls made from threads
      - Unexpected message
      - Crash



# Erlport and Tkinter Problems

-tKinter does not like multithreading!

-tKinter event loop needs to be on the main thread

-Event loop is infinite by design, program can’t receive messages

-Discovered late in development

-Solution? mtTkinter to the rescue!

# Evaluation and reflection

- Toy examples aren’t always enough
  - Weren’t able to determine conflicts until we had a fairly significant amount done
- More research before committing
  - Not getting surprised by library features (...or lack thereof!)
- Be more flexible
  - Stuck with erlport, should have investigated sockets

# Questions?