

デザインプロジェクトII

「ボードゲームAIの開発」

担当 小林・藤原

もくじ

開発環境の構築

どうぶつしょうぎサーバの体験

グループワーク

ティーチングアシスタント (TA)

長沼 一平

修士2年 | 小林研

所澤 亮太

修士1年 | 藤原研

主にコーディングする上での**ヒント**をくれます。
でも、決して**コーディング**は手伝いません。

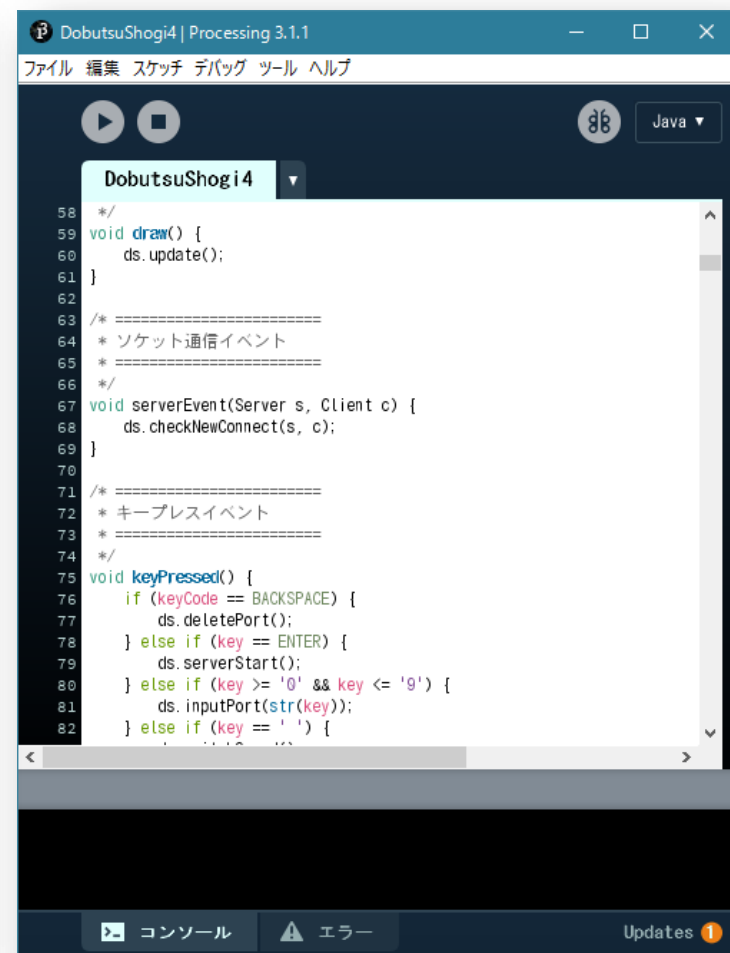
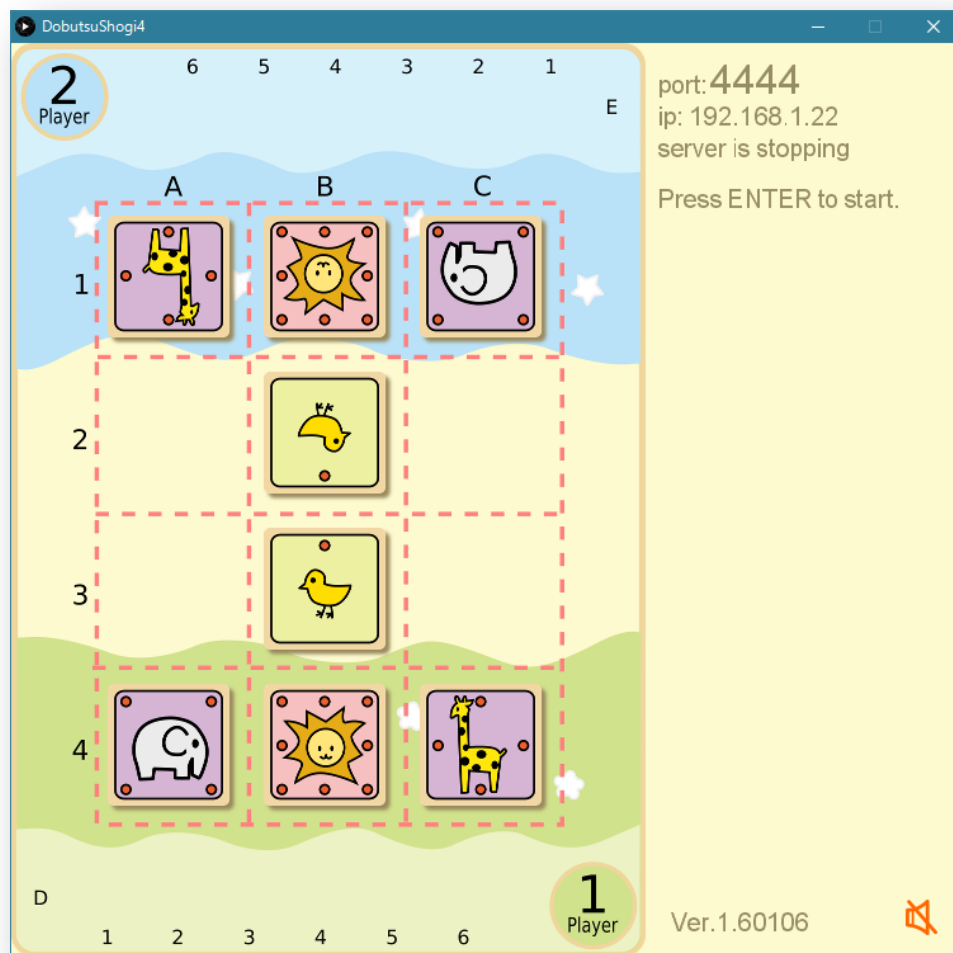
どうぶつしょうぎ

どうぶつしょうぎの対戦AIを開発する



どうぶつしょうぎ対戦可視化サーバ

どうぶつしょうぎの対戦AIを開発する



京都将棋

京都銀閣金鷄秘譜将棋

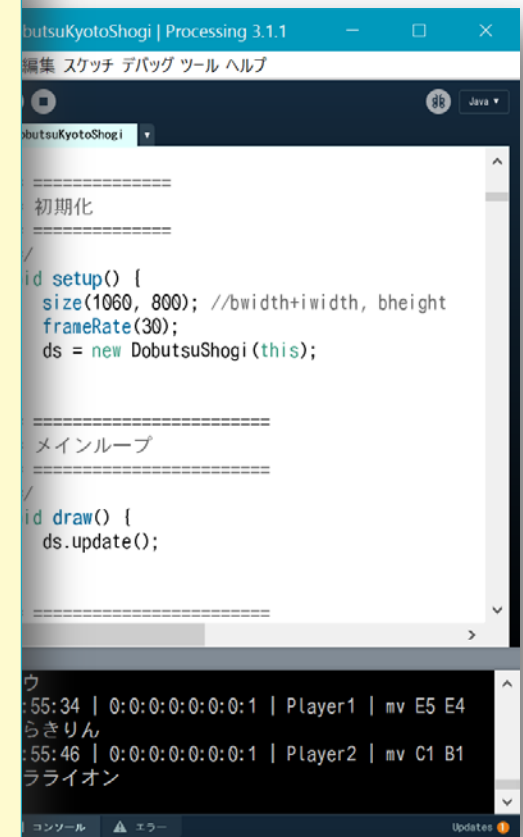
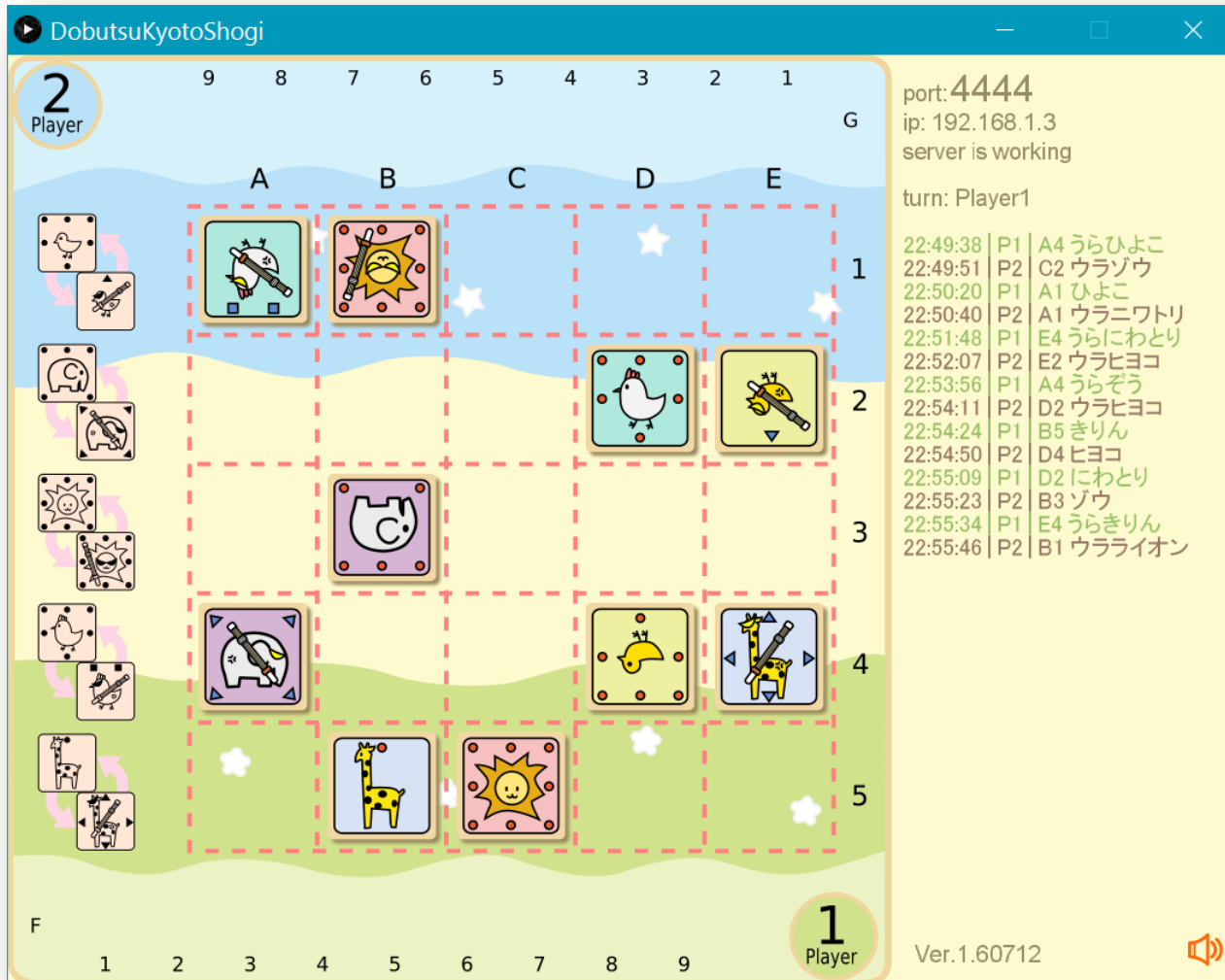
一手ごとにコマが裏返る5x5マスの将棋



玉 ⇄ 玉
香 ⇄ と
銀 ⇄ 角
金 ⇄ 桂
飛 ⇄ 歩

どうぶつきょうとしょうぎサーバ

京都将棋の対戦AIを開発する



どうぶつしょうぎAI

相手のAIに勝てる独自のAIを作るのが課題

```
tenum2.py (~/.Processing/DobutsuShogi2/Python) - GVIM
ファイル(E) 編集(E) ツール(T) シンタックス(S) パツファ(B) ウィンドウ(W) TeX-Suite TeX-Environments TeX-Ejements TeX-Math ヘルプ(H)

~/K/P/D/DobutsuShogi2.pde 3 /K/P/D/P/tenum2.py
1 #!/usr/bin/python
2 # -*- coding: utf-8 -*-
3
4 import socket
5 import sys
6 import re
7 import time
8 import threading
9 import itertools
10 import copy
11
12 BUFSIZE = 1024
13
14 # 1手=1つのインスタンス
15 class Te:
16
17     def __str__(self):
18         return "mv " + self.src + " " + self.dst
19
20     def __init__(self, teStr):
21         teStr = teStr.strip()
22
23         if teStr.split(" ")[0] != "mv":
24             raise ValueError("not mv")
25
26         self.src = teStr.split(" ")[1]
27         self.dst = teStr.split(" ")[2]
28
29 # 1つの局面=1つのインスタンス
30 # インスタンスが常に保持している情報は、辞書 masuToKoma のみ
31 class Kyokumen:
32
33     komadaiAlph = {"1": "D", "2": "E"}
34     flip = {"1": "2", "2": "1"}
35
36     # 駒の動き
37     # プレーヤー1を基準
38     kiki = [{"l": [(0, -1), (1, -1), (-1, -1), (1, 0), (-1, 0), (1, 1), (-1, 1), (0, -1)],
39              "e": [(1, -1), (-1, -1), (1, 1), (-1, 1)],
40              "g": [(0, -1), (1, 0), (-1, 0), (0, 1)],
41              "c": [(0, -1)],
42              "h": [(0, -1), (1, -1), (-1, -1), (1, 0), (-1, 0), (1, 1), (-1, 1)]}]
43
44 # 盤面簡易プリント用文字列
45 def debugStrings(self):
46     returnStr = "" * 20 + "\n\n"
47
48     # 持ち駒を文字列化
49     returnStr = returnStr + "Player2: "
50
51     for n in range(1,6):
52         komaStr = self.getMochigoma("2", n)
53
54         if komaStr != None:
55             returnStr = returnStr + komaStr + " "
56
57     returnStr = returnStr + "\n\n"
58
59     # 盤面を文字列化
60     for y in "1234":
61         for x in "ABCD":
62             komaStr = self.getKoma(x + y)
63
64             if komaStr == None:
65                 komaStr = "--"
66
67             returnStr = returnStr + komaStr
68
69     returnStr = returnStr + "\n"
70
71     # 持ち駒を文字列化
72     returnStr = returnStr + "Player1: "
73
74     for n in range(1,6):
75         komaStr = self.getMochigoma("1", n)
76
77         if komaStr != None:
78             returnStr = returnStr + komaStr + " "
79
80     returnStr = returnStr + "\n\n"
81
82     return returnStr
83
84 # 指定したマスにある駒の種類と現在の持ち主を返す
85 # 指定したマスに駒がなければ None
86 # マスの指定はXstrオブジェクトと整数を用いて行う
87 def getMochigoma(self, playerStr, n):
88     # 指定した手の持ち駒台の n 番目(1<=n<=6)にある駒の種類と現在の持ち主を返す
89     # 持ち駒台の n 番目(1<=n<=6)に駒がなければ None
90     # playerStr は "1" あるいは "2"
91     # 返り値は例えば "e1" という文字列
92     def getMochigoma(self, playerStr, n):
93         return self.masuToKoma.get(Kyokumen.komadaiAlph[playerStr] + str(n))
94
95     # 指定した手の持ち駒台の空いているところに駒を置く
96     # koma は例えば "e1" という文字列
97     def putMochigoma(self, playerStr, koma):
98         # 持ち駒台のうち、最初に空いているところ(つまり末尾)に置く
99         for n in range(1,7):
100             # 持ち駒台の n 番目に駒がなければそこに置く
101             if self.getMochigoma(playerStr, n) == None:
102                 self.masuToKoma[Kyokumen.komadaiAlph[playerStr] + str(n)] = koma
103                 break
104
105     # srcX, srcY のマス目の駒を動かす手を、プレーヤーがどちらかに関係なく生成するジェネレータ
106     # srcX, srcY に駒がなければエラー(呼び出し側でチェックすべし)
107     # 手とは、例えば ("B4", "C3") のようなタプル
108     def getTesFrom(self, src):
109         srcKoma = self.getKoma(src)
110
111         # プレーヤー2なら向き逆転
112         if srcKoma[1] == "1":
113             muki = 1
114         elif srcKoma[1] == "2":
115             muki = -1
116
117         # 駒の行先すべてについて
118         for (deltaX, deltaY) in Kyokumen.kiki[srcKoma[0]]:
119             deltaX = deltaX * muki
120             deltaY = deltaY * muki
121
122             # 行先が盤外の場合は飛ばす
123             if src[0] == "A" and deltaX < 0 or src[0] == "C" and deltaX > 0:
124                 continue
125
126             # 指定した手の持ち駒台の n 番目(1<=n<=6)にある駒の種類と現在の持ち主を返す
```


今日の授業の流れ

①開発環境の導入と説明

説明を聞きながら、一緒に作業する



②個人作業

Pythonプログラムからどうぶつしょうぎサーバに接続する



③グループ作業

- 個人作業を通して気づいた点や疑問点などを発表し合い、全員で考える
- 課題1-1(Pythonのディクショナリ), 1-2(どうぶつしょうぎサーバ) を考える



④リフレクションシートへの記入と提出

eALPSにアクセスし、今日の授業を振り返って各項目について記述する
課題1-1, 1-2 も忘れずに提出(次回の講義開始までに)

どうぶつしょうぎサーバ

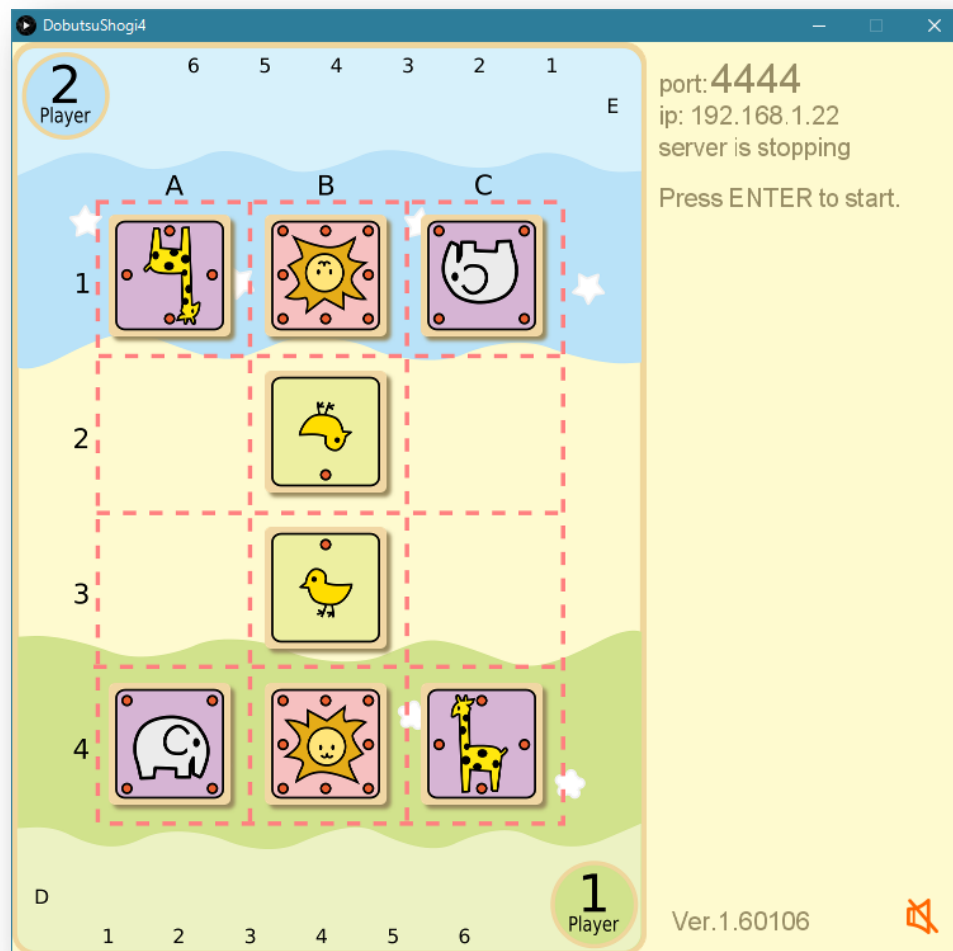
どうぶつしょうぎのコマの
動きを可視化するプログラム
(オリジナル)

クライアントからソケット通信
で接続してコマを動かす

マウスでも移動できる
(開発用, デバッグ用)

Mac:

```
xattr -rc DobutsuShogi5.app
```



どうぶつしょうぎサーバの仕様

制御すること

コマの表示

順番の管理

(先に接続した方がPlayer1)

自分のコマの上には置けない

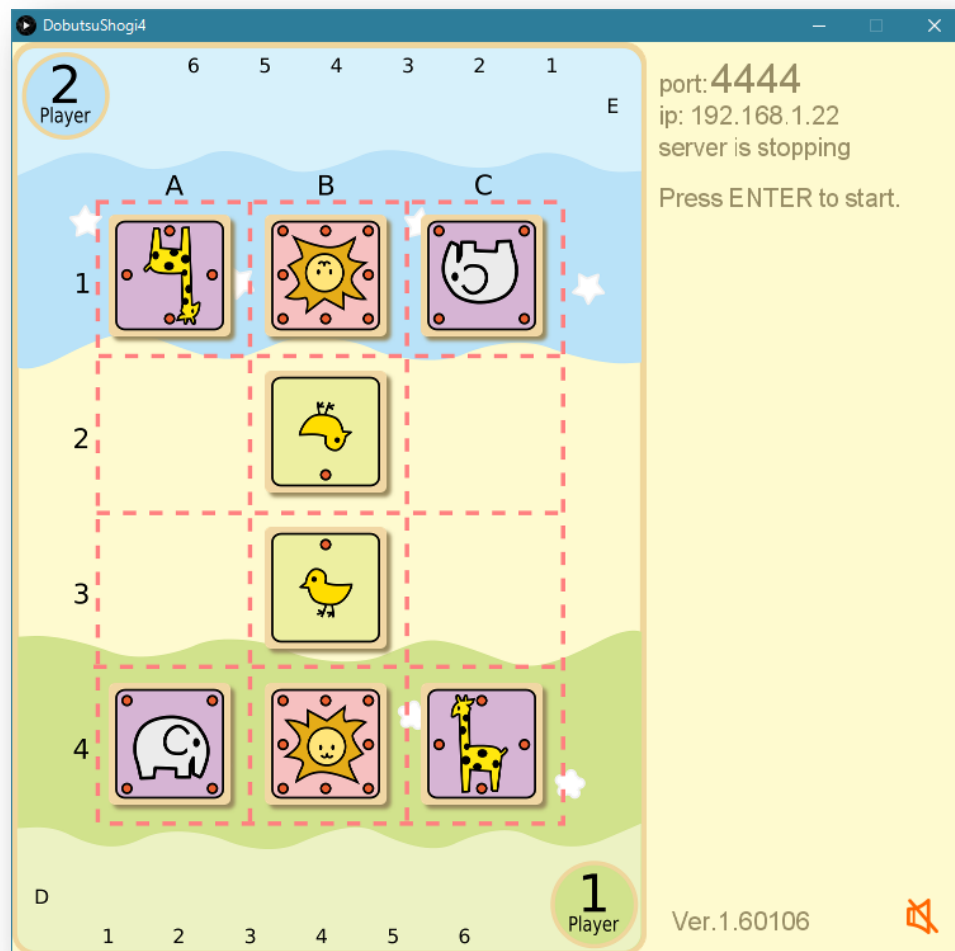
手駒は空白マスにしか置けない
自分のコマしか動かせない

制御しないこと

コマの移動可能範囲チェック

ルール違反

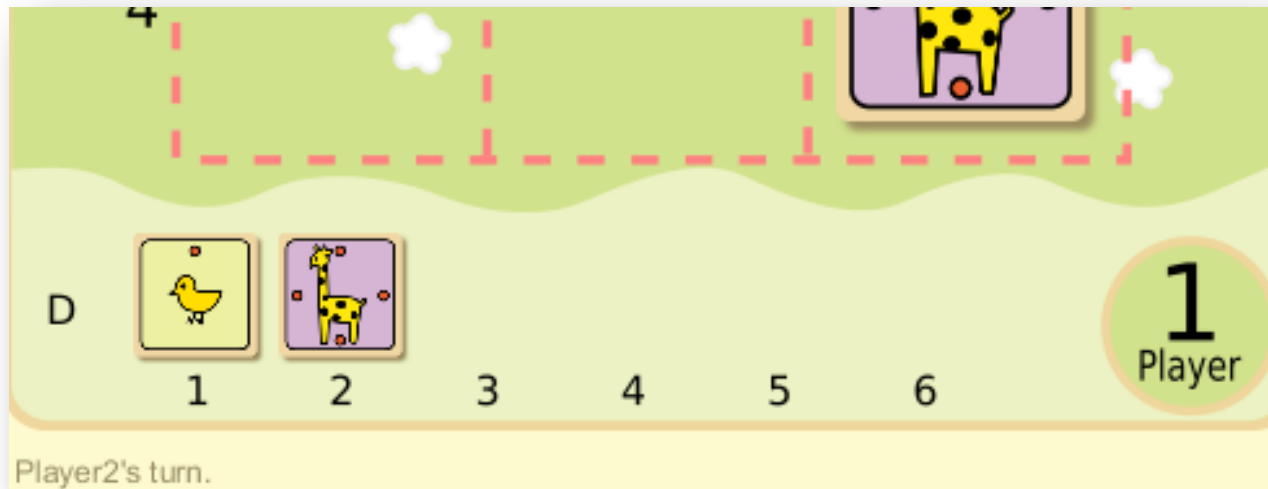
終局チェック



どうぶつしょうぎサーバの仕様

＜持ちゴマの挙動＞

- 持ちゴマを使うと数字の小さい方に詰まっていく



どうぶつしょうぎサーバの初回起動時

「アクセスを許可する」をクリック



手入力でどうぶつしょうぎ

Python3

ソケット通信
クライアント

```
import socket
import re
import time

BUFSIZE = 1024

serverName = "localhost"
serverPort = 4444

s = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
s.connect((serverName, serverPort))
print(s.recv(BUFSIZE).rstrip().decode())

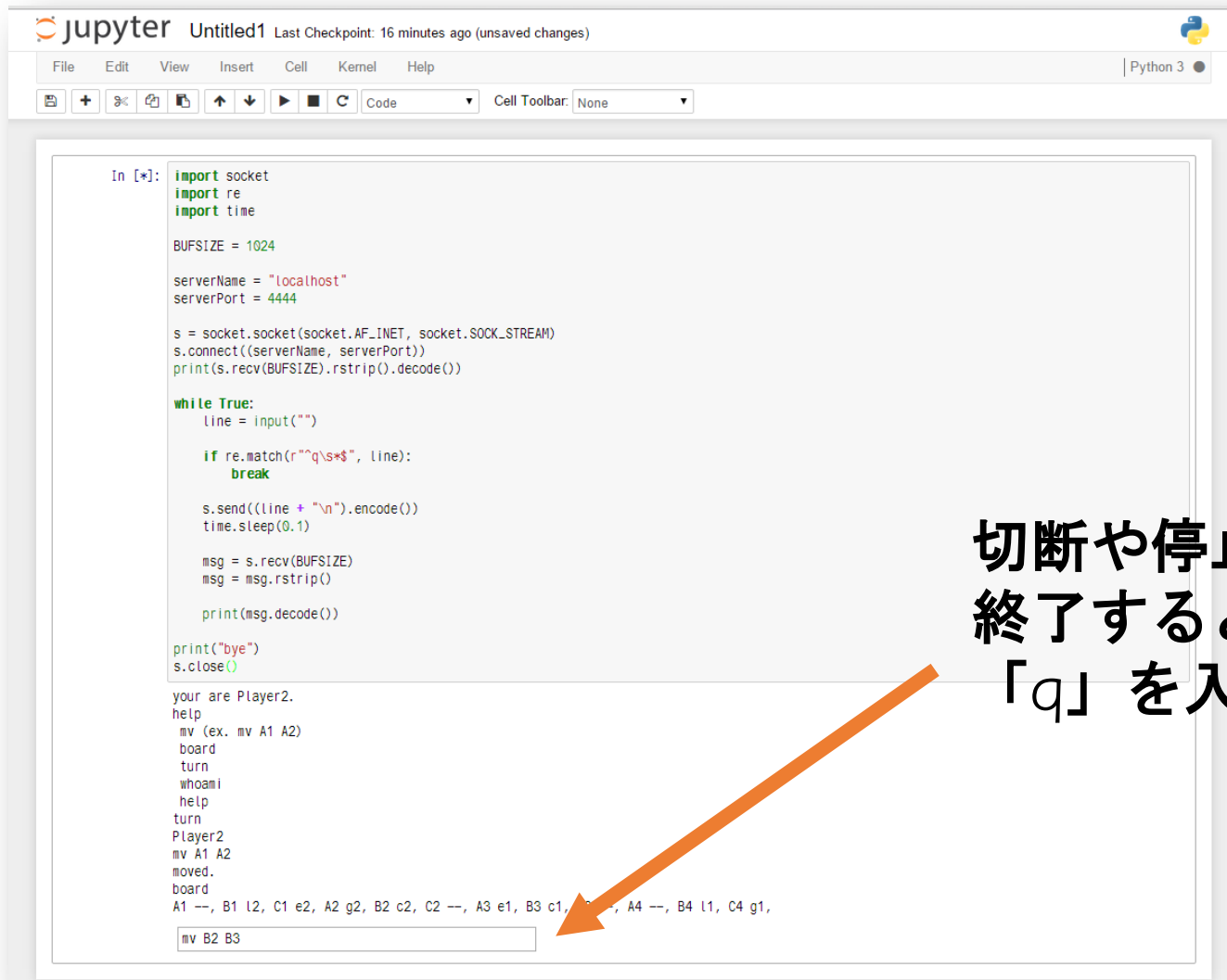
while True:
    line = input("")

    if re.match(r"^\q\fs*$", line):
        break

    s.send((line + "\n").encode())
    time.sleep(0.1)

print(s.recv(BUFSIZE).rstrip().decode())
```

手入力でどうぶつしょうぎ



```
In [*]: import socket
import re
import time

BUFSIZE = 1024

serverName = "localhost"
serverPort = 4444

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((serverName, serverPort))
print(s.recv(BUFSIZE).rstrip().decode())

while True:
    line = input("")

    if re.match(r"^[q\s]+$", line):
        break

    s.send((line + "\n").encode())
    time.sleep(0.1)

    msg = s.recv(BUFSIZE)
    msg = msg.rstrip()

    print(msg.decode())

print("bye")
s.close()

your are Player2.
help
mv (ex. mv A1 A2)
board
turn
whoami
help
turn
Player2
mv A1 A2
moved.
board
A1 --, B1 l2, C1 e2, A2 g2, B2 c2, C2 --, A3 e1, B3 c1, A4 --, A4 --, B4 l1, C4 g1,

mv B2 B3
```

切断や停止など,
終了するとき必ず
「q」を入力すること

どうぶつしょうぎサーバへの接続

serverName = "localhost"

**をサーバを動作させている
PCのIPアドレスに変更する
(他者のPCに接続する)**

例)

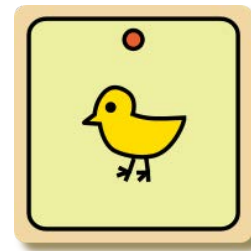
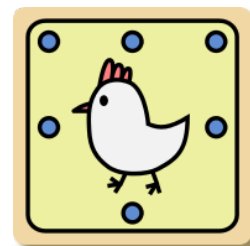
serverName = "10.2.70.130"

**表示されているIPが192.168.56.1の場合は
cmd.exeからipconfigして正しいIPを取得する**

どうぶつしょうぎのルール

<基本的な情報>

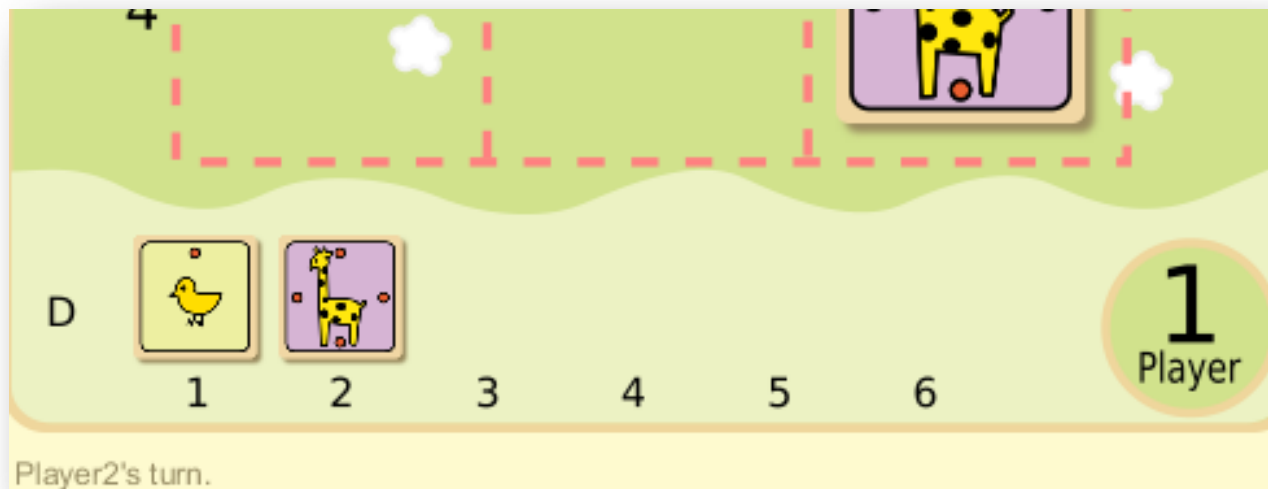
- 2人で順番にコマを動かして遊ぶゲーム
- コマはライオン，ぞう，きりん，ひよこの4つ
- コマの上に描かれた点の方向に1マス進める



どうぶつしょうぎのルール

<コマの動かし方>

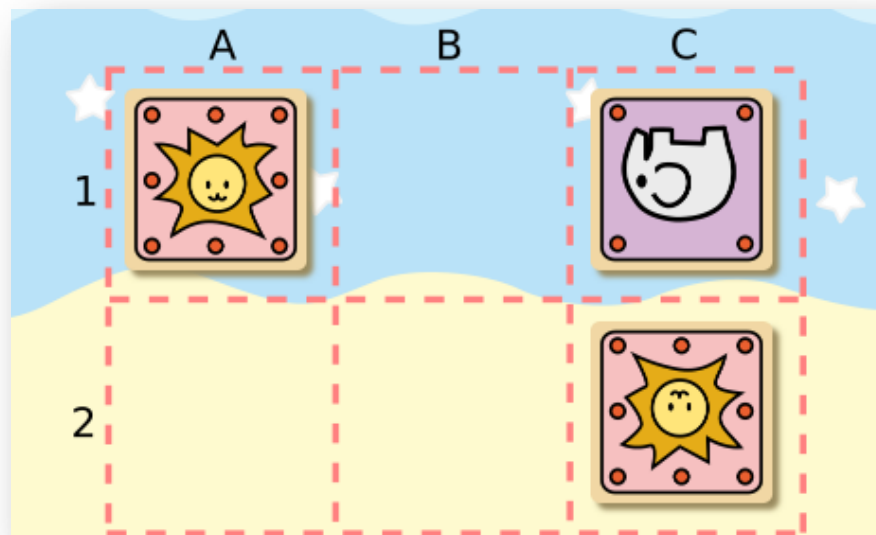
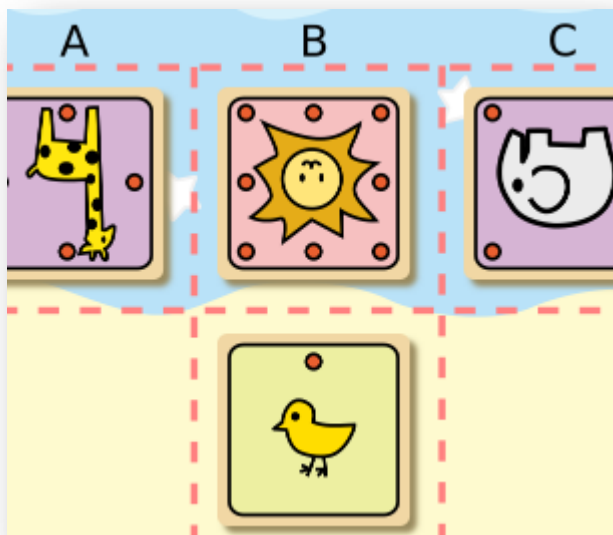
- 1つのマスに2匹の動物は置けない
- 相手の動物がいるマスに，自分の動物を進めると捕まえることができる．盤上から取って手元に置く(持ち駒)
- 持ち駒は，自分の番のときに空いているマスに置ける



どうぶつしょうぎのルール

<勝敗>

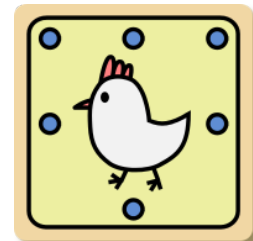
- 相手のライオンを先に捕まえた方が勝ち（キャッチ）
- 自分のライオンを、相手エリアの端まで先に進めた方が勝ち（トライ）
ただし、次に相手に捕まえてしまう場合は負け



どうぶつしょうぎのルール

＜その他＞

- ひよこは、相手の一番奥まで行くとニワトリになる
- 相手のニワトリをとっても、使うときはひよこから相手エリアの一番奥に置いても、ひよこ
- 同じ場面が3回出現したら引き分け
- パスはなし
- 将棋との違い
二歩（2ひよこ）、打ち歩詰め（打ちひよこ詰め）、
行き所のないコマ打ち、連続王手の千日手は反則にならない



手入力はどうぶつしょうぎ

ソケット通信クライアントを使った入力対戦

- 対戦相手とペアになる
- どうぶつしょうぎサーバをダウンロード
- どちらか一方がサーバを動作させる
- 表示されたIPとポート番号に接続する
- 先に接続した方が先手となる
- 「help」で命令一覧を習得 (全部で6つ)
- mv, board, initboard, turn, whoami, help
- コマを動かすには, 「mv A4 A3」のように, 座標を指定必ずアルファベットが先になるように

マウスでどうぶつしょうぎ

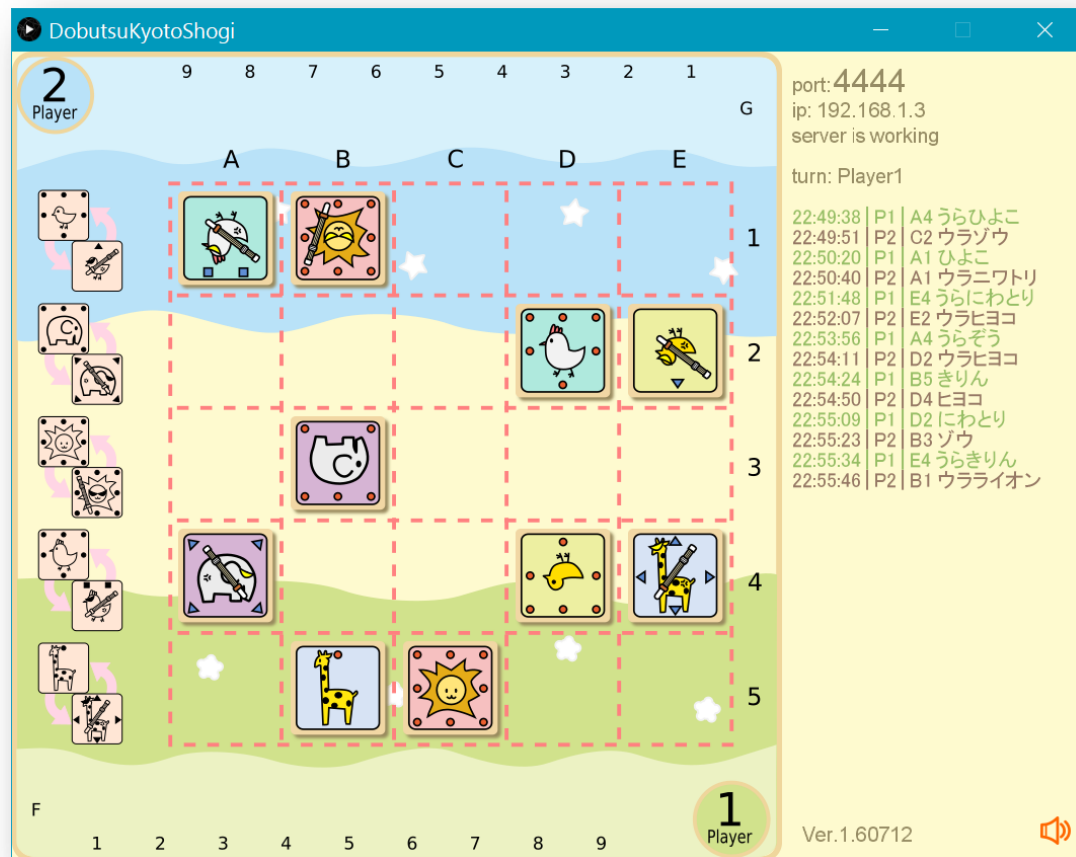
- マウスで操作したい場合
クライアント接続するかわりに「I」を押す
- 例) Player1はソケット通信, Player2をマウス操作
 - ① 「waiting Player1...」と表示されているときに,
ソケット通信でサーバに接続する
 - ② Player1が接続されると表示が「waiting Player2...」
にかわるので, キーボードの「I」を押す
- 動かしたいコマをクリック
→ 置きたいところでもう一度クリック (ドラッグは不可)
- 両者ともマウス操作することも可能
- 移動をキャンセルしたときは, 元の場所に配置する

きょうとしょうぎサーバ

京都将棋のコマを
どうぶつしょうぎ化し、
動きを可視化する
プログラム（オリジナル）

クライアントから
ソケット通信で接続して
コマを動かす

マウスでも移動できる
（開発用、デバッグ用）



きょうとしょうぎサーバの仕様

制御すること

コマの表示

順番の管理

(先に接続した方がPlayer1)

自分のコマの上には置けない

手駒は空白マスにしか置けない

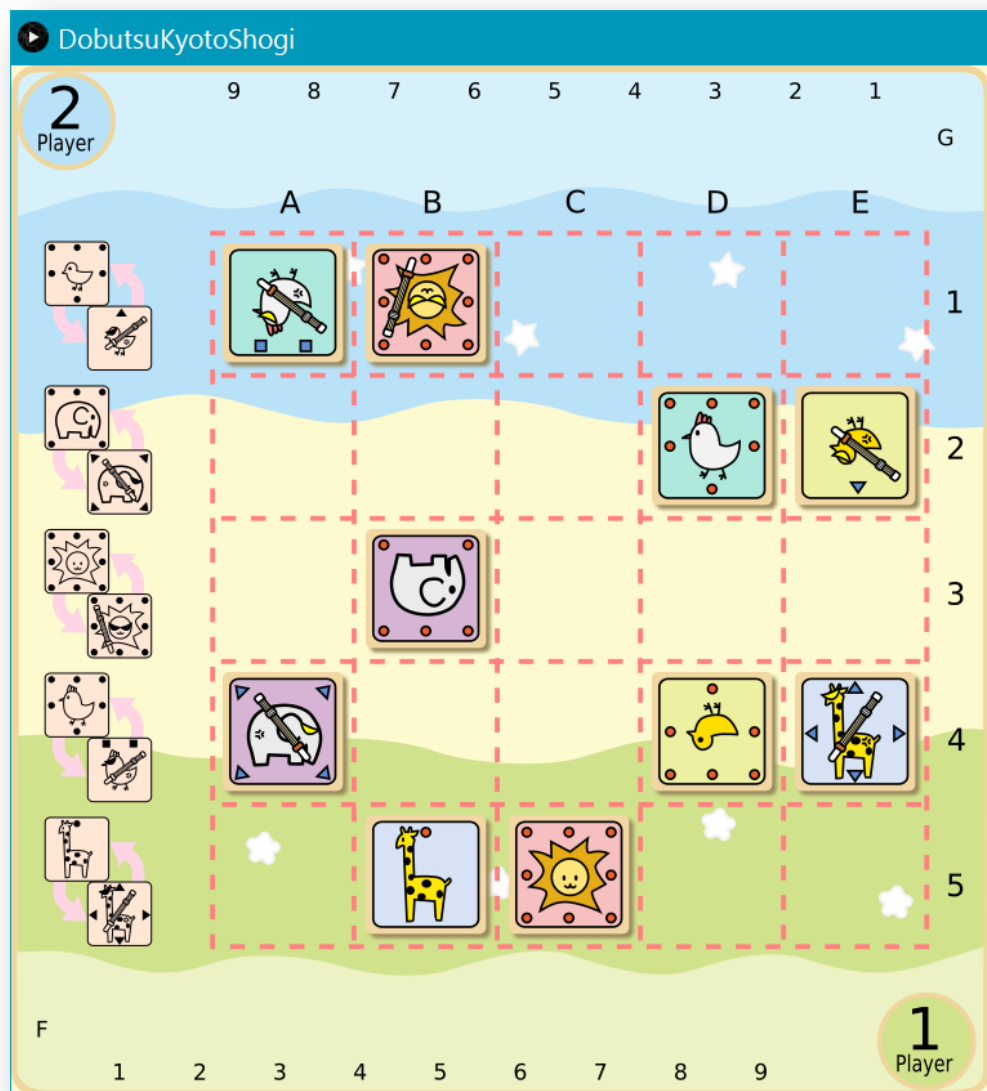
自分のコマしか動かせない

制御しないこと

コマの移動可能範囲チェック

ルール違反

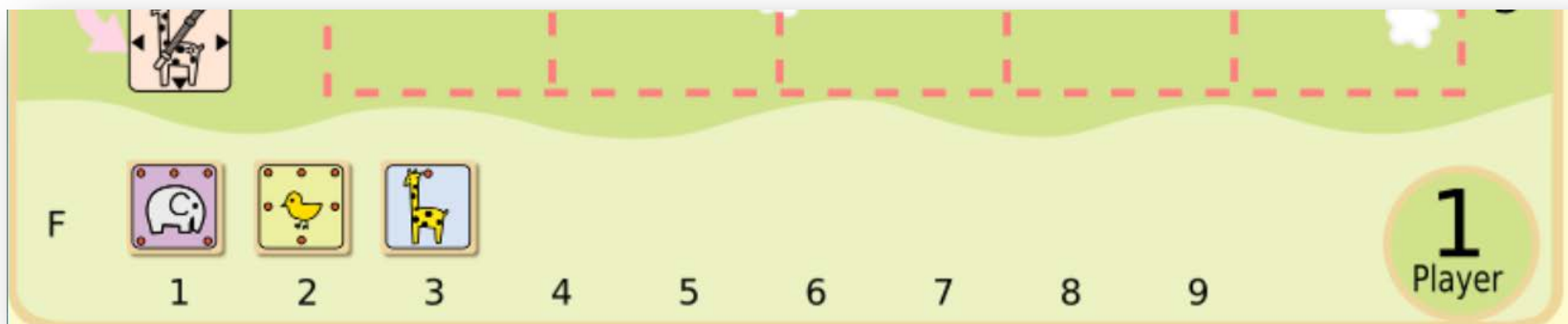
終局チェック



きょうとしょうぎサーバの仕様

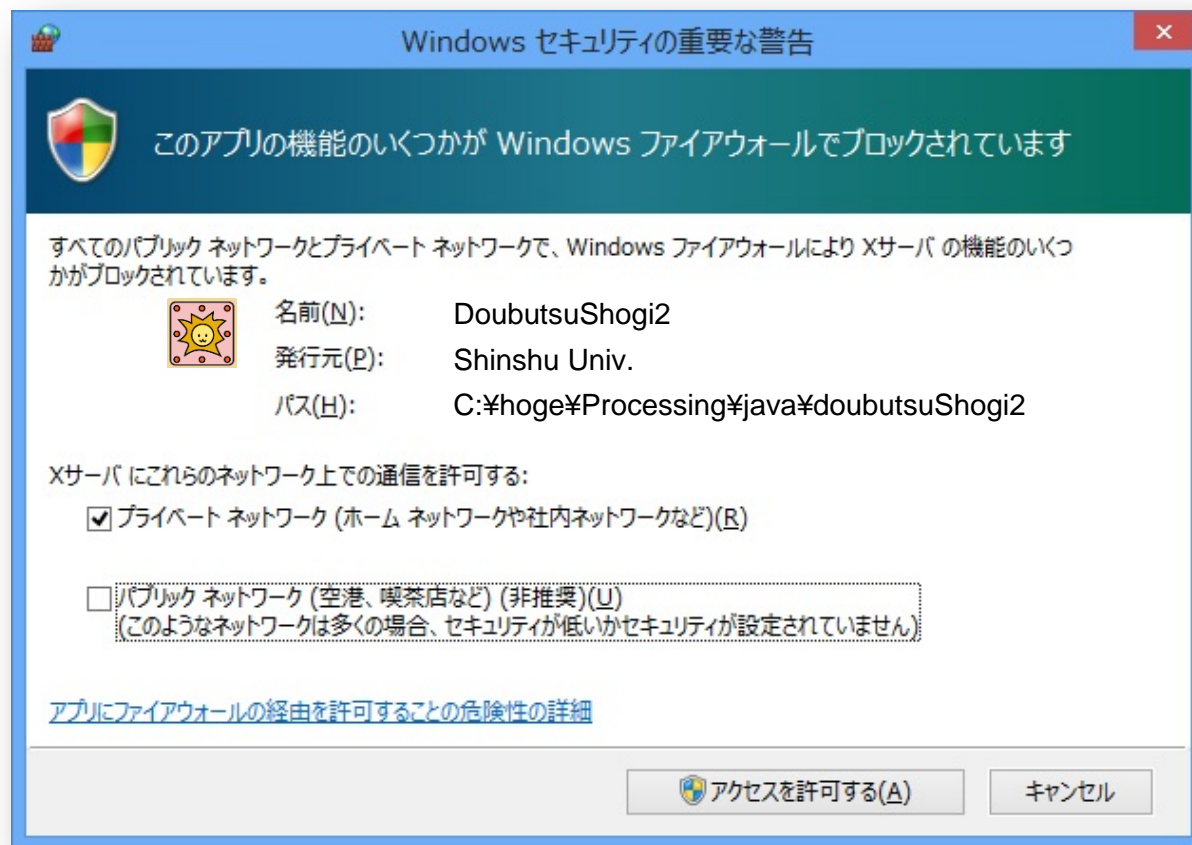
＜持ちゴマの挙動＞

- 持ちゴマを使うと数字の小さい方に詰まっていく



サーバの初回起動時

「アクセスを許可する」をクリック



手入力できょうとしょうぎ

Python3 ソケット通信 クライアント

```
import socket
import re
import time

BUFSIZE = 1024

serverName = "localhost"
serverPort = 4444

s = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
s.connect((serverName, serverPort))
print(s.recv(BUFSIZE).rstrip().decode())

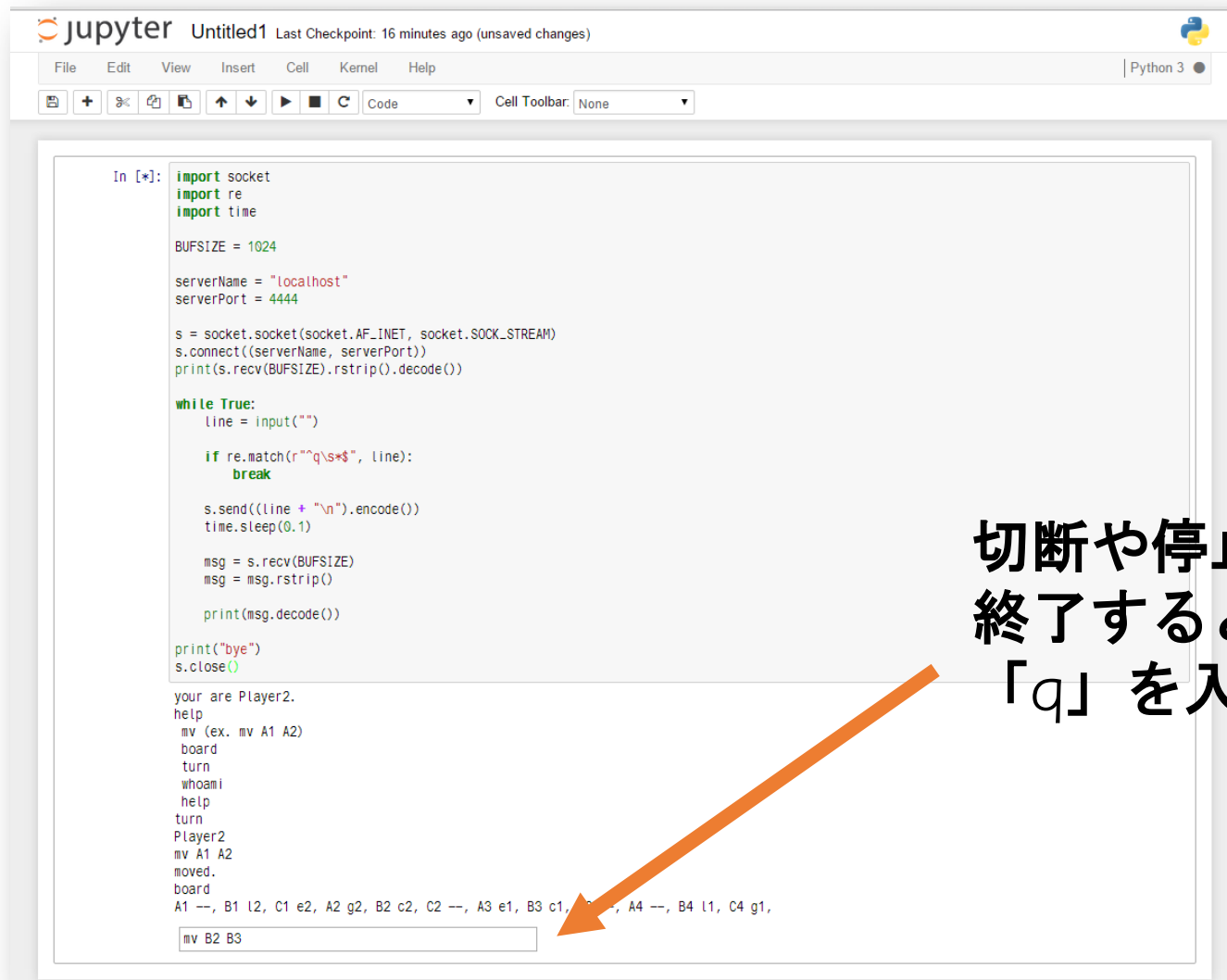
while True:
    line = input("")

    if re.match(r"^\q\fs*$", line):
        break

    s.send((line + "\n").encode())
    time.sleep(0.1)

print(s.recv(BUFSIZE).rstrip().decode())
```

手入力できょうとしょうぎ



```
In [*]: import socket
import re
import time

BUFSIZE = 1024

serverName = "localhost"
serverPort = 4444

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((serverName, serverPort))
print(s.recv(BUFSIZE).rstrip().decode())

while True:
    line = input("")

    if re.match(r"^[q\s]+$", line):
        break

    s.send((line + "\n").encode())
    time.sleep(0.1)

    msg = s.recv(BUFSIZE)
    msg = msg.rstrip()

    print(msg.decode())

print("bye")
s.close()

your are Player2.
help
mv (ex. mv A1 A2)
board
turn
whoami
help
turn
Player2
mv A1 A2
moved.
board
A1 --, B1 l2, C1 e2, A2 g2, B2 c2, C2 --, A3 e1, B3 c1, A4 --, A4 --, B4 l1, C4 g1,

mv B2 B3
```

切断や停止など,
終了するとき必ず
「q」を入力すること

サーバへの接続

serverName = "localhost"

**をサーバを動作させている
PCのIPアドレスに変更する
(他者のPCに接続する)**

例)

serverName = "10.2.70.130"

**表示されているIPが192.168.56.1の場合は
cmd.exeからipconfigして正しいIPを取得する**

きょうとしょうぎのルール

＜基本的な情報＞

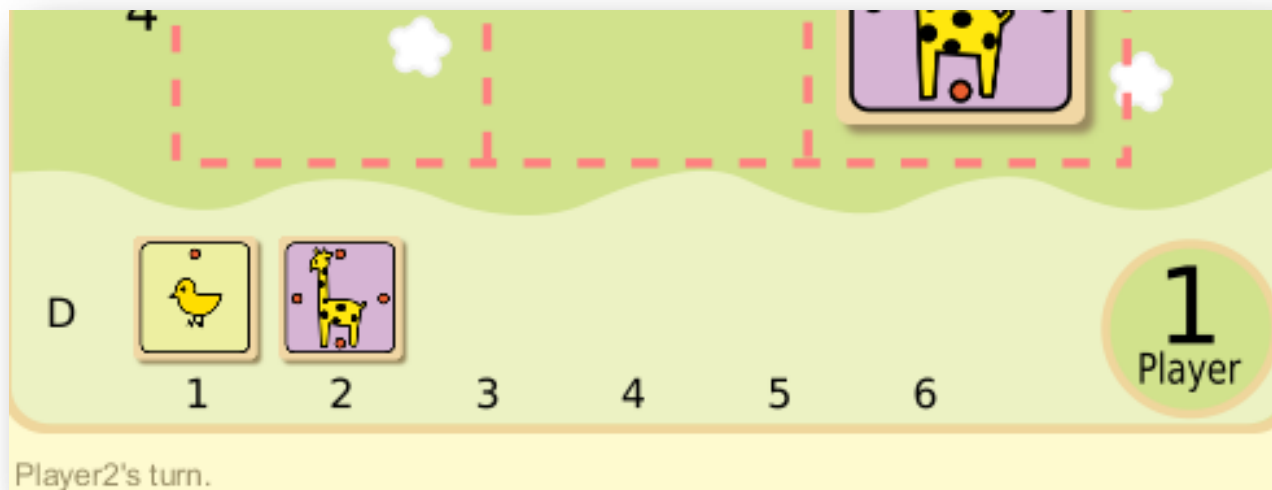
- 2人で順番にコマを動かして遊ぶゲーム（普通の将棋がベース）
- コマを1つ動かすごとに裏返る
- コマはライオン，ぞう，きりん，にわとり，ひよこの5つ
- コマの上に描かれた図形に従って移動できる
赤い丸点：その方向に1マス進める
青い三角：その方向に好きな数だけ進める
青い四角：1つ前に進む＋斜め（桂馬の動き）



きょうとしょうぎのルール

<コマの動かし方>

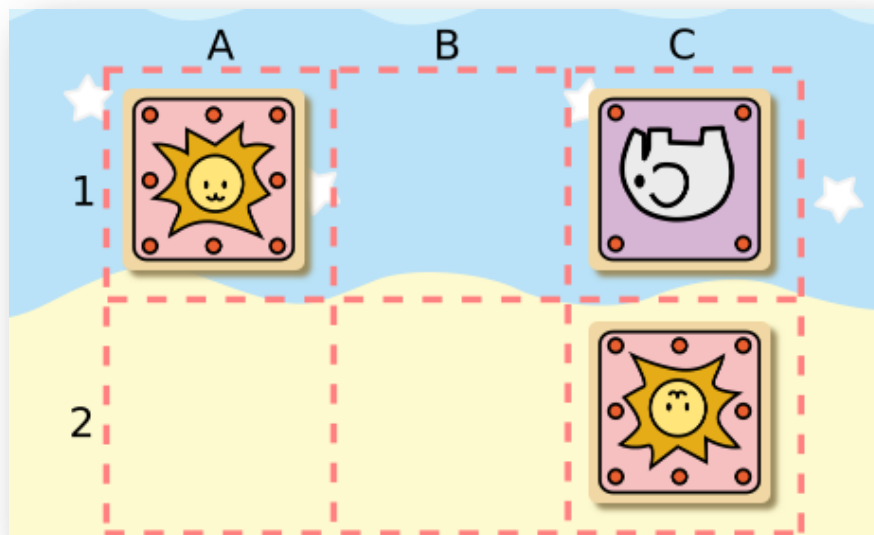
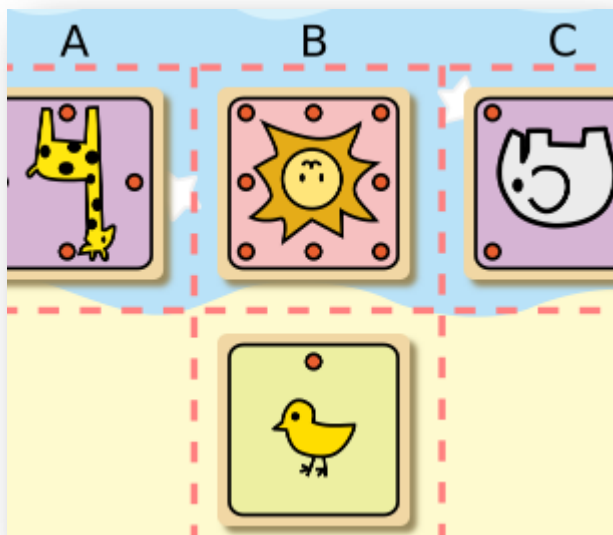
- 1つのマスに2匹の動物は置けない
- 相手の動物がいるマスに，自分の動物を進めると捕まえることができる．盤上から取って手元に置く(持ち駒)
- 持ち駒は，自分の番のときに空いているマスに置ける
- 持ち駒は表，裏，どちらにしてもよい



きょうとしょうぎのルール

＜勝敗＞

- 相手のライオンを先に捕まえた方が勝ち
- どうぶつしょうぎのトライは適用されない



きょうとしょうぎのルール

＜その他＞

- パスはなし
- 将棋との違い
二歩（二きりん），打ち歩詰め（打ちきりん詰め），
行き所のないコマ打ち
は反則にならない
- 千日手は引き分け

手入力できょうとしょうぎ

ソケット通信クライアントを使ったて入力対戦

- 対戦相手とペアになる
- どうぶつしょうぎサーバをダウンロード
- どちらか一方がサーバを動作させる
- 表示されたIPとポート番号に接続する
- 先に接続した方が先手となる
- 「help」で命令一覧を習得 (全部で7つ)
- mv, mvl, board, initboard, turn, whoami, help
- コマを動かすには, 「mv A4 A3」のように, 座標を指定必ずアルファベットが先になるように
- mvlを使うと, 持ち駒を裏返して配置できる

マウスできょうとしょうぎ

- マウスで操作したい場合
クライアント接続するかわりに「I」を押す
- 例) Player1はソケット通信, Player2をマウス操作
 - ① 「waiting Player1...」と表示されているときに,
ソケット通信でサーバに接続する
 - ② Player1が接続されると表示が「waiting Player2...」
にかわるので, キーボードの「I」を押す
- 動かしたいコマをクリック
→ 置きたいところでもう一度クリック (ドラッグは不可)
- 両者ともマウス操作することも可能
- 移動をキャンセルしたときは, 元の場所に配置する
- 持ち駒を反転したい場合には,
コマを選択してから右クリックする

スケジュール

日付	内容
10/3	ガイダンス
10/10	開発環境の構築・グループワーク・コーディング
10/17	グループワーク・コーディング
10/24	公式対戦会 その1
10/31	グループワーク・コーディング
11/7	公式対戦会 その2
11/14	グループワーク・コーディング
11/28	公式対戦会 その3

やること

① 個人作業

どうぶつしょうぎサーバにプログラムから
接続してコマを動かしてみる

Pythonを学習するのによい情報源を探す
気づいた点や理解する上でのコツ,
疑問点などをメモする

② グループ作業

2人～3人のグループをつくり,
互いに気づいた点やコツを発表する
全員発表したら, 続いて,
疑問点を発表し, 全員で解決する.

やること(続き)

③リフレクションシートへの記入

1. グループ作業に対する自己評価
☐一人で実施した ☐教えてもらった ☐教えた
☐しっかり聴いた ☐質問した
2. 今日の授業内容のまとめ
3. 解決できなかった疑問
4. 感想・意見など

グループ作業で意識すること

- しっかり聴く
- よく考える
- 質問する
- グループへの貢献
- 批判・非難をしない
- 積極的な参加

開発手順のヒント

- 盤面データを、ディクショナリとして保存
- ルールにそった手をすべて生成する
 - 移動元と移動先のタプルを列挙
- 盤面データ上でコマを移動する
- 1手読みして指す
- n 手読みして指す
- 高速化

アンケート

- どうぶつしょうぎをやりたい
- きょうとしょうぎをやりたい
- どうぶつしょうぎのプログラムなら作れそう