

c0a2400443 / ProjExD\_Group09

🔍

📧

👤

<> Code

🔗 Pull requests

🎬 Actions

📁 Projects

📖 Wiki

⋮

# Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#) or [learn more about diff comparisons](#).

🔗

base: main

←

compare: C0A24004/King

Discuss and review the changes in this comparison with others. [Learn about pull requests](#)

Create pull request

🔗 2 commits

📄 1 file changed

👤 1 contributor

🔗 Commits on Jul 8, 2025

キングの移動を実装

📄 8b77f40

<>

c0a2400443 committed 1 hour ago

ゲームの終了を実装

📄 464175f

<>

c0a2400443 committed 18 minutes ago

📄 Showing 1 changed file with 37 additions and 4 deletions.

Split Unified

▼ 41 chess.py

1

1

import pygame

2

2

import sys

3

3

from enum import Enum

4

4

5

5

# 初期化

6

6

pygame.init()

7

7

8

8

# 定数

9

9

BOARD\_SIZE = 8

10

10

SQUARE\_SIZE = 80

11

11

WINDOW\_WIDTH = BOARD\_SIZE \* SQUARE\_SIZE

12

12

WINDOW\_HEIGHT = BOARD\_SIZE \* SQUARE\_SIZE + 100 # 情報表示用のスペース

13

13

FPS = 60

14

14

15

15

# 色の定義

16

16

WHITE = (255, 255, 255)

17

17

BLACK = (0, 0, 0)

18

18

LIGHT\_BROWN = (240, 217, 181)

19

19

DARK\_BROWN = (181, 136, 99)

20

20

HIGHLIGHT\_COLOR = (255, 255, 0, 128)

21

21

SELECTED\_COLOR = (0, 255, 0, 128)

22

22

RED = (255, 0, 0)

23	23	BLUE = (0, 0, 255)
24	24	
25	25	class PieceType(Enum):
26	26	PAWN = 1
27	27	ROOK = 2
28	28	KNIGHT = 3
29	29	BISHOP = 4
30	30	QUEEN = 5
31	31	KING = 6
32	32	
33	33	class PieceColor(Enum):
34	34	WHITE = 1
35	35	BLACK = 2
36	36	
37	37	class Piece:
38	38	def __init__(self, piece_type, color, row, col):
39	39	self.type = piece_type
40	40	self.color = color
41	41	self.row = row
42	42	self.col = col
43	43	self.has_moved = False
44	44	
45	45	def get_possible_moves(self, board):
46	46	"""駒の可能な動きを取得（現在は全方向移動可能）"""
47	47	moves = []
48	-	for row in range(8):
49	-	for col in range(8):
50	-	if board.is_valid_move(self.row, self.col, row, col):
51	-	moves.append((row, col))
	48	+ if self.type == PieceType.KING:
	49	+ directions = [(-1, -1), (-1, 0), (-1, 1), (0, -1), (0, 1), (1, -1), (1, 0),
		(1, 1)]
	50	+ for dr, dc in directions:
	51	+ new_row = self.row + dr
	52	+ new_col = self.col + dc
	53	+ if 0 <= new_row < 8 and 0 <= new_col < 8:
	54	+ target_piece = board.get_piece(new_row, new_col)
	55	+ if not target_piece or target_piece.color != self.color:
	56	+ moves.append((new_row, new_col))
	57	+ else:
	58	+ for row in range(8):
	59	+ for col in range(8):
	60	+ if board.is_valid_move(self.row, self.col, row, col):
	61	+ moves.append((row, col))
52	62	return moves
53	63	
54	64	def move(self, new_row, new_col):
55	65	"""駒を移動"""
56	66	self.row = new_row
57	67	self.col = new_col
58	68	self.has_moved = True
59	69	
60	70	def __str__(self):
61	71	symbols = {
62	72	PieceType.PAWN: "P",
63	73	PieceType.ROOK: "R",
64	74	PieceType.KNIGHT: "N",
65	75	PieceType.BISHOP: "B",

66	76	<code>PieceType.QUEEN: "Q",</code>
67	77	<code>PieceType.KING: "K"</code>
68	78	<code>}</code>
69	79	<code>return symbols[self.type]</code>
70	80	
71	81	<code>def get_display_color(self):</code>
72	82	<code>    """駒の表示色を取得"""</code>
73	83	<code>    return BLACK if self.color == PieceColor.BLACK else WHITE</code>
74	84	
75	85	<code>class ChessBoard:</code>
76	86	<code>    def __init__(self):</code>
77	87	<code>        self.board = [[None for _ in range(8)] for _ in range(8)]</code>
78	88	<code>        self.current_turn = PieceColor.WHITE</code>
79	89	<code>        self.selected_piece = None</code>
80	90	<code>        self.selected_pos = None</code>
	91	<code>+ self.winner = None</code>
81	92	<code>        self.possible_moves = []</code>
82	93	<code>        self.setup_initial_position()</code>
83	94	
84	95	<code>    def setup_initial_position(self):</code>
85	96	<code>        """初期配置を設定"""</code>
86	97	<code>        # 黒の駒</code>
87	98	<code>        piece_order = [PieceType.ROOK, PieceType.KNIGHT, PieceType.BISHOP,</code>
		<code>PieceType.QUEEN,</code>
88	99	<code>                        PieceType.KING, PieceType.BISHOP, PieceType.KNIGHT, PieceType.ROOK]</code>
89	100	
90	101	<code>        for col in range(8):</code>
91	102	<code>            # 黒の駒</code>
92	103	<code>            self.board[0][col] = Piece(piece_order[col], PieceColor.BLACK, 0, col)</code>
93	104	<code>            self.board[1][col] = Piece(PieceType.PAWN, PieceColor.BLACK, 1, col)</code>
94	105	
95	106	<code>            # 白の駒</code>
96	107	<code>            self.board[7][col] = Piece(piece_order[col], PieceColor.WHITE, 7, col)</code>
97	108	<code>            self.board[6][col] = Piece(PieceType.PAWN, PieceColor.WHITE, 6, col)</code>
98	109	
99	110	<code>    def get_piece(self, row, col):</code>
100	111	<code>        """指定位置の駒を取得"""</code>
101	112	<code>        if 0 &lt;= row &lt; 8 and 0 &lt;= col &lt; 8:</code>
102	113	<code>            return self.board[row][col]</code>
103	114	<code>        return None</code>
104	115	
105	116	<code>    def set_piece(self, row, col, piece):</code>
106	117	<code>        """指定位置に駒を配置"""</code>
107	118	<code>        if 0 &lt;= row &lt; 8 and 0 &lt;= col &lt; 8:</code>
108	119	<code>            self.board[row][col] = piece</code>
109	120	
110	121	<code>    def is_valid_move(self, from_row, from_col, to_row, to_col):</code>
111	122	<code>        """移動が有効かチェック（基本的な範囲チェック）"""</code>
112	123	<code>        if not (0 &lt;= to_row &lt; 8 and 0 &lt;= to_col &lt; 8):</code>
113	124	<code>            return False</code>
114	125	
115	126	<code>        piece = self.get_piece(from_row, from_col)</code>
116	127	<code>        if not piece:</code>
117	128	<code>            return False</code>
118	129	
119	130	<code>        target_piece = self.get_piece(to_row, to_col)</code>
120	131	<code>        if target_piece and target_piece.color == piece.color:</code>
121	132	<code>            return False</code>

122	133	
123	134	<code>return True</code>
124	135	
125	136	<code>def make_move(self, from_row, from_col, to_row, to_col):</code>
126	137	<code>    """駒を移動"""</code>
127	138	<code>    piece = self.get_piece(from_row, from_col)</code>
128	139	
129	140	<code>    # 基本的な移動可能性チェック</code>
130	141	<code>    if not piece:</code>
131	142	<code>        return False</code>
132	143	
133	144	<code>    # 現在のターンの駒かチェック</code>
134	145	<code>    if piece.color != self.current_turn:</code>
135	146	<code>        return False</code>
136	147	
137	148	<code>    # 移動先が有効かチェック</code>
138	149	<code>    if not self.is_valid_move(from_row, from_col, to_row, to_col):</code>
139	150	<code>        return False</code>
	151	<code>+</code>
	152	<code>+</code>
	153	<code>    #target_pieceを定義</code>
	154	<code>+</code>
	155	<code>    # キングが取られたら勝敗を設定</code>
	156	<code>    if target_piece and target_piece.type == PieceType.KING:</code>
	157	<code>        self.set_piece(to_row, to_col, piece)</code>
	158	<code>        self.set_piece(from_row, from_col, None)</code>
	159	<code>        piece.move(to_row, to_col)</code>
	160	<code>        self.winner = piece.color # 勝った側の色</code>
	161	<code>        return True</code>
	162	<code>+</code>
140	163	
141	164	<code>    # 移動実行</code>
142	165	<code>    self.set_piece(to_row, to_col, piece)</code>
143	166	<code>    self.set_piece(from_row, from_col, None)</code>
144	167	<code>    piece.move(to_row, to_col)</code>
145	168	
146	169	<code>    # ターン切り替え</code>
147	170	<code>    self.current_turn = PieceColor.BLACK if self.current_turn == PieceColor.WHITE else</code>
		<code>    PieceColor.WHITE</code>
148	171	<code>    return True</code>
149	172	
150	173	<code>def select_piece(self, row, col):</code>
151	174	<code>    """駒を選択"""</code>
152	175	<code>    piece = self.get_piece(row, col)</code>
153	176	<code>    if piece and piece.color == self.current_turn:</code>
154	177	<code>        self.selected_piece = piece</code>
155	178	<code>        self.selected_pos = (row, col)</code>
156	179	<code>        self.possible_moves = piece.get_possible_moves(self)</code>
157	180	<code>        return True</code>
158	181	<code>    return False</code>
159	182	
160	183	<code>def deselect_piece(self):</code>
161	184	<code>    """駒の選択を解除"""</code>
162	185	<code>    self.selected_piece = None</code>
163	186	<code>    self.selected_pos = None</code>
164	187	<code>    self.possible_moves = []</code>
165	188	
166	189	<code>class ChessGame:</code>

```

167 190     def __init__(self):
168 191         self.screen = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT))
169 192         pygame.display.set_caption("チェスゲーム")
170 193         self.clock = pygame.time.Clock()
171 194         self.board = ChessBoard()
172 195         # フォントの設定（日本語対応）
173 196         try:
174 197             self.font = pygame.font.Font("msgothic.ttc", 24) # Windows
175 198         except:
176 199             try:
177 200                 self.font = pygame.font.Font("NotoSansCJK-Regular.ttc", 24) # Linux
178 201             except:
179 202                 self.font = pygame.font.Font(None, 24) # フォールバック
180 203
181 204         self.piece_font = pygame.font.Font(None, 60)
182 205
183 206     def get_board_pos(self, mouse_pos):
184 207         """マウス位置をボード座標に変換"""
185 208         x, y = mouse_pos
186 209         if 0 <= x < WINDOW_WIDTH and 0 <= y < BOARD_SIZE * SQUARE_SIZE:
187 210             col = x // SQUARE_SIZE
188 211             row = y // SQUARE_SIZE
189 212             return row, col
190 213         return None, None
191 214
192 215     def draw_board(self):
193 216         """チェスボードを描画"""
194 217         for row in range(BOARD_SIZE):
195 218             for col in range(BOARD_SIZE):
196 219                 color = LIGHT_BROWN if (row + col) % 2 == 0 else DARK_BROWN
197 220                 rect = pygame.Rect(col * SQUARE_SIZE, row * SQUARE_SIZE, SQUARE_SIZE,
SQUARE_SIZE)
198 221                 pygame.draw.rect(self.screen, color, rect)
199 222
200 223                 # 選択中のマスハイライト
201 224                 if self.board.selected_pos == (row, col):
202 225                     highlight_surface = pygame.Surface((SQUARE_SIZE, SQUARE_SIZE),
pygame.SRCALPHA)
203 226                     highlight_surface.fill(SELECTED_COLOR)
204 227                     self.screen.blit(highlight_surface, (col * SQUARE_SIZE, row *
SQUARE_SIZE))
205 228
206 229                 # 可能な移動先をハイライト
207 230                 if (row, col) in self.board.possible_moves:
208 231                     highlight_surface = pygame.Surface((SQUARE_SIZE, SQUARE_SIZE),
pygame.SRCALPHA)
209 232                     highlight_surface.fill(HIGHLIGHT_COLOR)
210 233                     self.screen.blit(highlight_surface, (col * SQUARE_SIZE, row *
SQUARE_SIZE))
211 234
212 235     def draw_pieces(self):
213 236         """駒を描画"""
214 237         for row in range(BOARD_SIZE):
215 238             for col in range(BOARD_SIZE):
216 239                 piece = self.board.get_piece(row, col)
217 240                 if piece:
218 241                     # 駒のテキストを描画
219 242                     text_color = piece.get_display_color()

```

220	243	# 背景色を設定（見やすくするため）
221	244	bg_color = WHITE if text_color == BLACK else BLACK
222	245	
223	246	text = self.piece_font.render(str(piece), True, text_color)
224	247	text_rect = text.get_rect(center=(col * SQUARE_SIZE + SQUARE_SIZE //
	248	2,
225	248	row * SQUARE_SIZE + SQUARE_SIZE // 2))
226	249	
227	250	# 背景の円を描画
228	251	pygame.draw.circle(self.screen, bg_color, text_rect.center, 25)
229	252	pygame.draw.circle(self.screen, text_color, text_rect.center, 25, 2)
230	253	
231	254	self.screen.blit(text, text_rect)
232	255	
233	256	def draw_info(self):
234	257	"""ゲーム情報を描画"""
235	258	info_y = BOARD_SIZE * SQUARE_SIZE + 10
	259	+
	260	+
	261	if self.board.winner != None:
	262	winner_color = "White" if self.board.winner == PieceColor.WHITE else "Black"
	263	result_text = f"{winner_color} wins!"
	264	text = self.font.render(result_text, True, RED)
	265	self.screen.blit(text, (10, info_y))
	266	return # 勝敗が決まったら他の表示は不要
236	266	
237	267	# 現在のターン（英語で表示）
238	268	turn_text = f"Current Turn: {'White' if self.board.current_turn ==
	269	PieceColor.WHITE else 'Black'}"
239	269	text = self.font.render(turn_text, True, BLACK)
240	270	self.screen.blit(text, (10, info_y))
241	271	
242	272	# 選択中の駒（英語で表示）
243	273	if self.board.selected_piece:
244	274	piece = self.board.selected_piece
245	275	color_name = "White" if piece.color == PieceColor.WHITE else "Black"
246	276	selected_text = f"Selected: {color_name} {str(piece)} at ({piece.row},
	277	{piece.col})"
247	277	text = self.font.render(selected_text, True, BLACK)
248	278	self.screen.blit(text, (10, info_y + 30))
249	279	
250	280	def handle_click(self, mouse_pos):
251	281	"""マウスクリックを処理"""
	282	+
	283	if self.board.winner:
	284	return # 勝敗が決まったらクリック操作無効
252	285	row, col = self.get_board_pos(mouse_pos)
253	286	if row is not None and col is not None:
254	287	if self.board.selected_piece:
255	288	# 駒が選択されている場合
256	289	if (row, col) in self.board.possible_moves:
257	290	# 有効な移動先がクリックされた場合
258	291	from_row, from_col = self.board.selected_pos
259	292	if self.board.make_move(from_row, from_col, row, col):
260	293	print(f"Move made: {from_row},{from_col} -> {row},{col}")
261	294	print(f"Turn changed to: {self.board.current_turn}")
262	295	self.board.deselect_piece()
263	296	else:
264	297	print("Move failed")

```
265 298         elif self.board.select_piece(row, col):
266 299             # 別の駒を選択（現在のターンの駒のみ）
267 300             print(f"Selected piece: {self.board.selected_piece} at ({row},
{col}))")
268 301         else:
269 302             # 無効な場所がクリックされた場合、選択解除
270 303             self.board.deselect_piece()
271 304             print("Deselected piece")
272 305         else:
273 306             # 駒が選択されていない場合
274 307             if self.board.select_piece(row, col):
275 308                 print(f"Selected piece: {self.board.selected_piece} at ({row},
{col}))")
276 309             else:
277 310                 print(f"Cannot select piece at ({row}, {col})")
278 311
279 312     def run(self):
280 313         """メインゲームループ"""
281 314         running = True
282 315         while running:
283 316             for event in pygame.event.get():
284 317                 if event.type == pygame.QUIT:
285 318                     running = False
286 319                 elif event.type == pygame.MOUSEBUTTONDOWN:
287 320                     if event.button == 1: # 左クリック
288 321                         self.handle_click(event.pos)
289 322
290 323             # 描画
291 324             self.screen.fill(WHITE)
292 325             self.draw_board()
293 326             self.draw_pieces()
294 327             self.draw_info()
295 328
296 329             pygame.display.flip()
297 330             self.clock.tick(FPS)
298 331
299 332             pygame.quit()
300 333             sys.exit()
301 334
302 335     # メイン実行
303 336     if __name__ == "__main__":
304 337         game = ChessGame()
305 338         game.run()
```