Pivit

Mestrado Integrado em Engenharia Informática e Computação

Inteligência Artificial

Faculdade de Engenharia da Universidade do Porto

Grupo Pivit:

Pedro Leite Galvão - 201700488

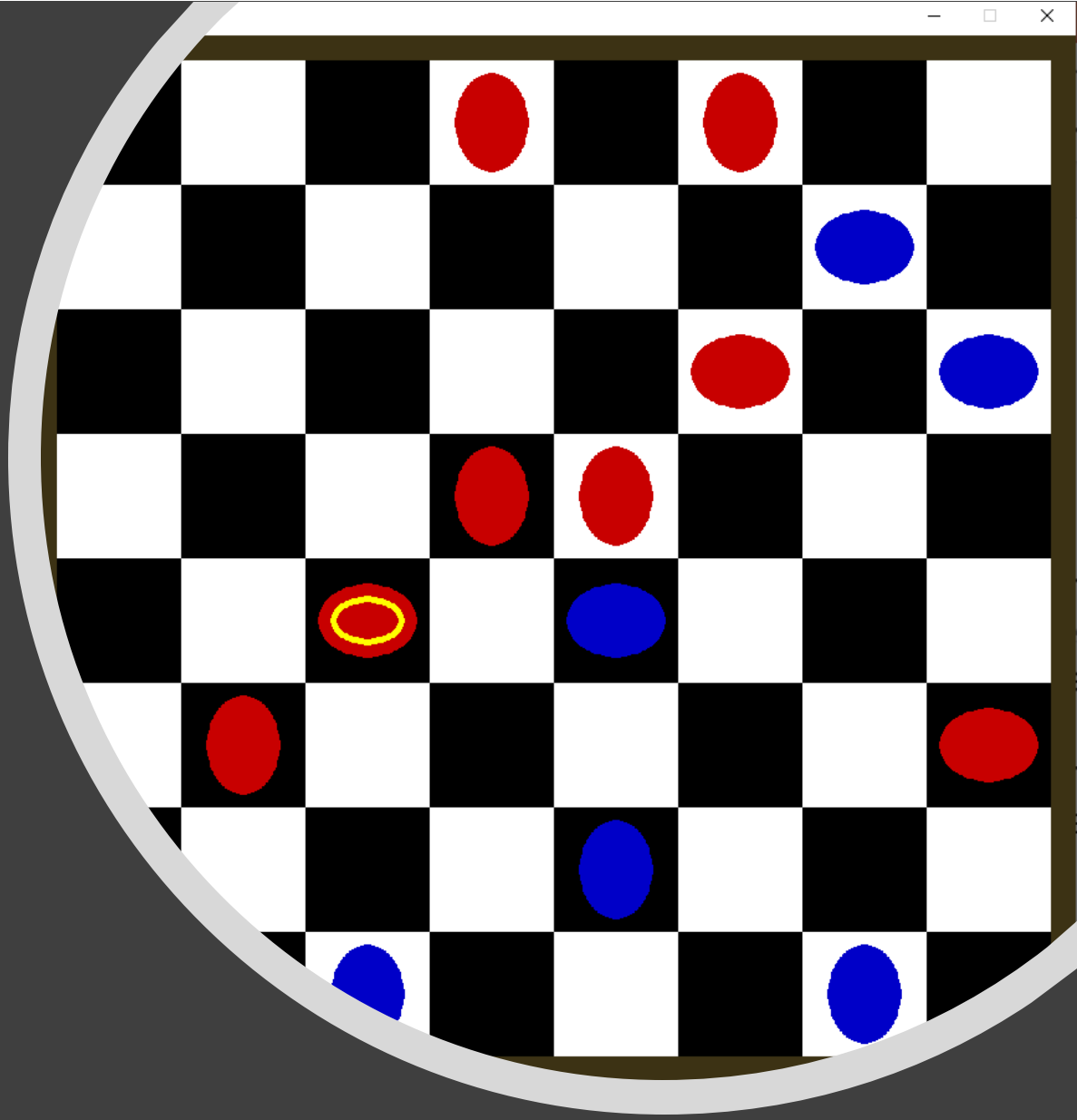
Vasco Marques Lopes Teixeira - 201802112

Decrição do Jogo

O nosso trabalho centra-se na replicação do jogo de tabuleiro “Pivit”, que é um jogo de tabuleiro para dois jogadores.

O campo de jogo é um tabuleiro quadrado como o de xadrez, 8x8 ou 6x6. As peças podem capturar-se umas às outras como no xadrez e ser promovidas de *minions* a *masters*.

Cada peça tem na sua face setas que apontam na direcção em que se pode mover – horizontal ou vertical. Todas as vezes que uma peça se move, ela gira 90 graus no sítio onde ficou, mudando de direcção

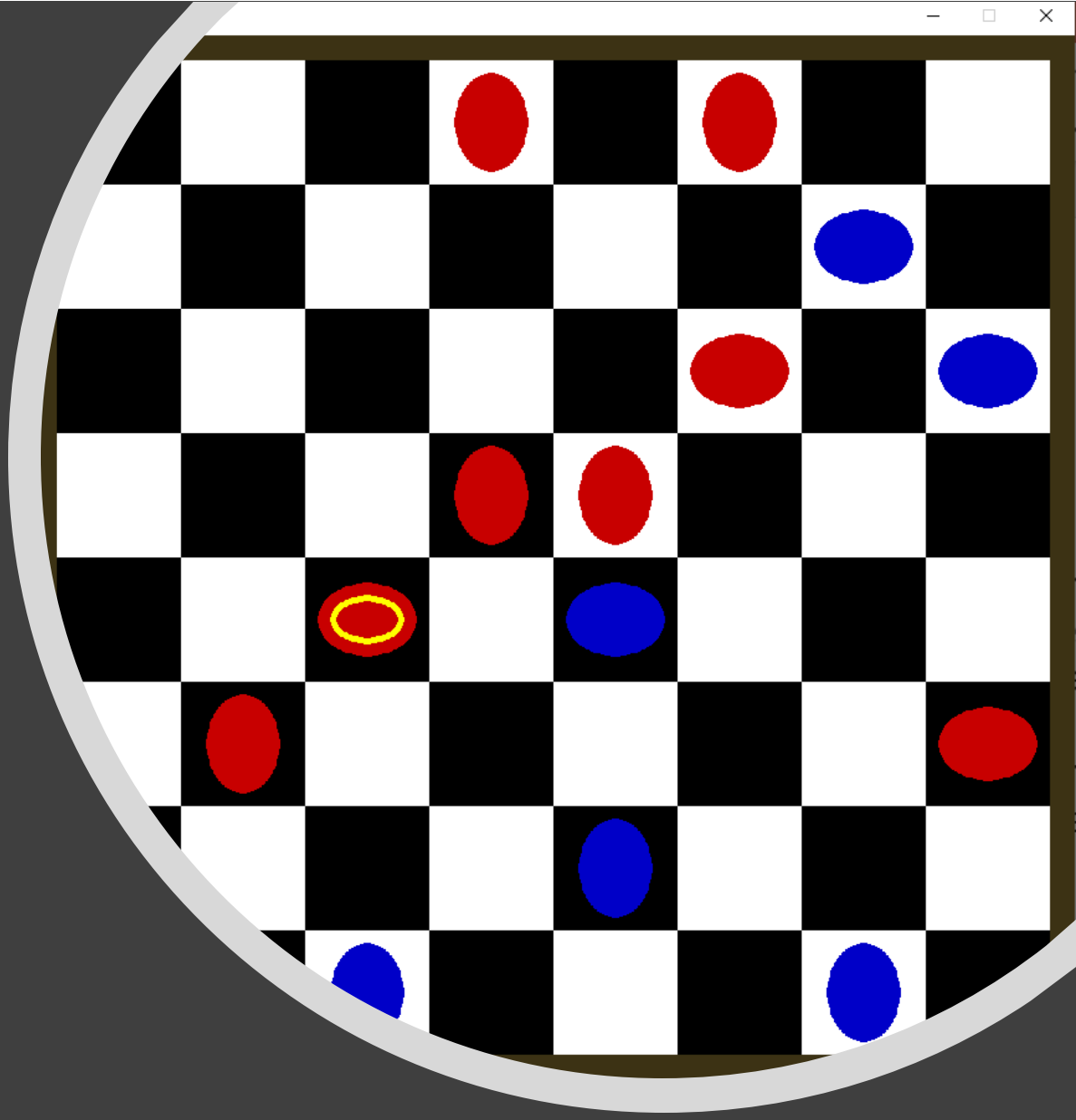


Descrição do Jogo

As peças básicas, chamadas de *minions*, devem movimentar-se para um quadrado de uma cor diferente. Ou seja, se eles começam num quadrado preto, devem movimentar-se para um quadrado branco. Peças promovidas, chamadas *masters*, podem pousar em qualquer lugar ao longo da sua linha.

As peças não podem saltar por cima de outras e capturam ao aterrar numa casa ocupada pelo inimigo. A promoção ocorre quando um *minion* se movimenta para um espaço do canto do tabuleiro.

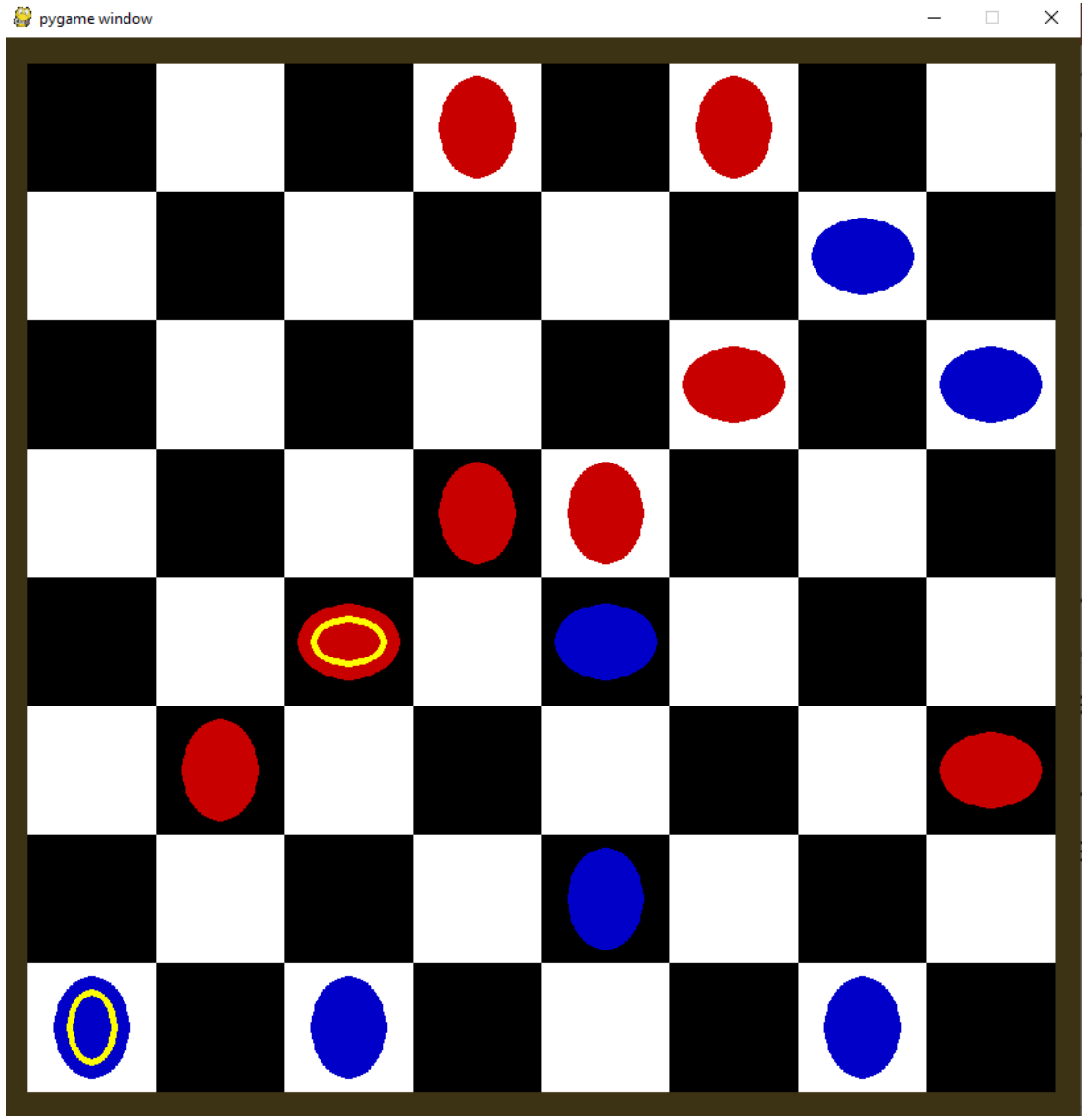
O jogo termina quando todos os *minions* de ambos os jogadores são capturados. O vencedor é o jogador com mais *masters* no tabuleiro.



Formulação do Problema

- **<u>Representação do estado:</u>**

O jogo é representado por um tabuleiro com peças nas casas das extremidades menos nos cantos. Para isso usamos uma lista de listas para representar o tabuleiro em que cada índice representa uma casa. Dentro de cada posição da lista pode estar 0 que representa uma casa vazia, ou um objecto “Piece” que representa uma peça. Utilizamos também uma variável “active_player” booleana que muda a cada movimento para determinar qual o jogador a jogar.



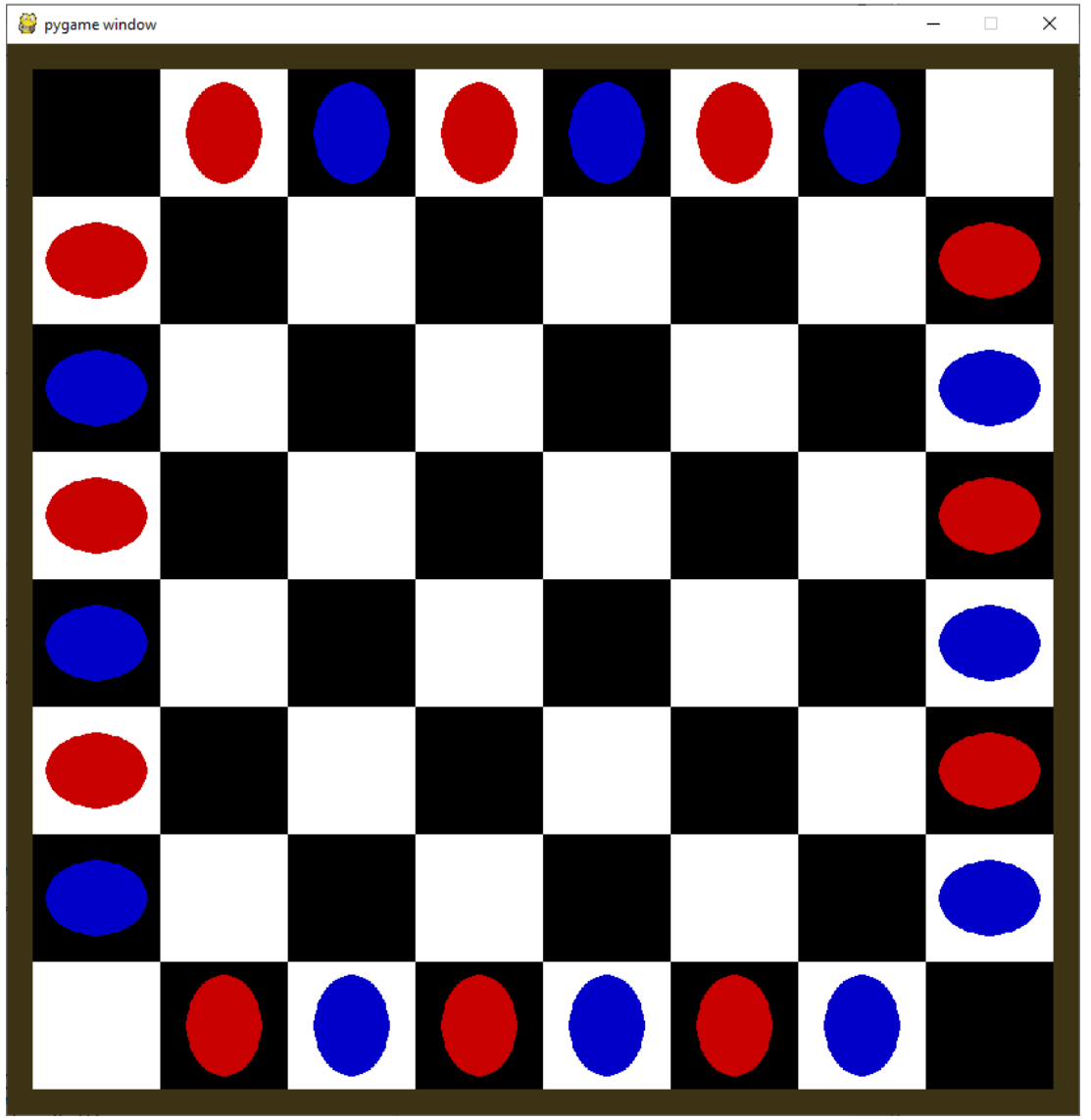
Formulação do Problema

- **Estado inicial:**

No estado inicial as peças dos dois jogadores ficam colocadas nas linhas e colunas encostadas às extremidades. As peças são colocadas alternadamente pela cor e os cantos do tabuleiro ficam vazios assim como todas as restantes casas.

- **Operadores:**

As possíveis operações a serem realizadas a partir de um estado são determinadas seguindo as regras de jogo já descritas. Dado que há muitos movimentos possíveis, estes operadores e seus efeitos devem ser determinados a cada jogada. Não existem custos envolvidos nas jogadas.



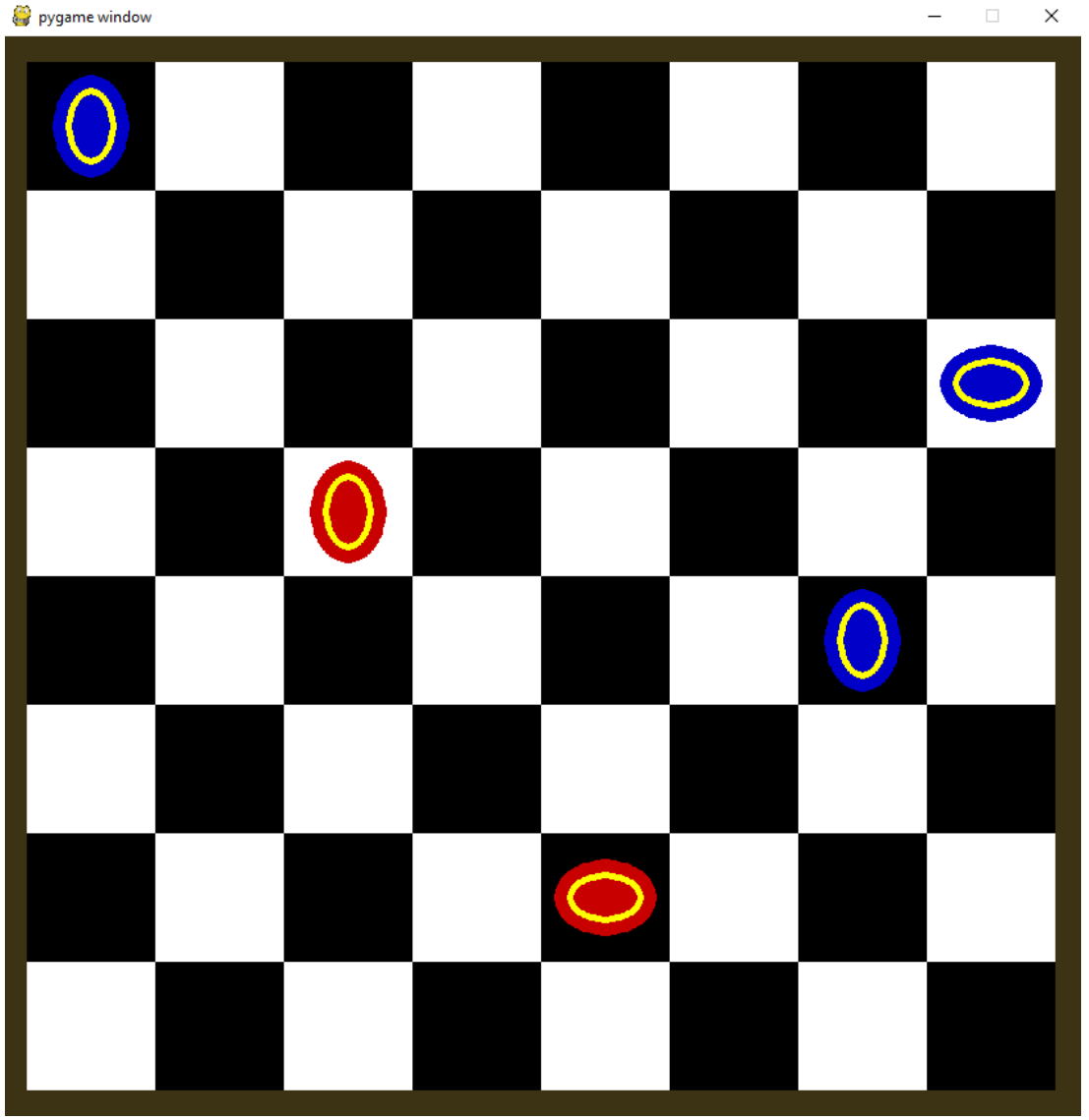
Formulação do Problema

- **Teste Terminal:**

A condição para o jogo terminar é que haja apenas masters no tabuleiro. Vence o jogador com mais masters.

- **Função de Avaliação:**

A função de avaliação implementamos faz a soma dos valores de todas peças de um jogador e subtrai pelas do oponente. Os minions valem 1 ponto e os masters valem 2.



Implementação

Já temos implementada em python uma interface gráfica com um menu inicial e um modo de jogo para dois jogadores humanos.

As funcionalidades necessárias a isto estão implementadas no ficheiro pivit.py na classe Game.

Começamos a fazer em um ficheiro separado uma classe que deverá ser responsável por escolher os movimentos do computador utilizando algoritmo minmax, cortes alpha-beta e o que mais considerarmos necessário.

```
7  import pygame
8  from random import *
9  from time import sleep
10 from time import time
11
12
13 class Piece(object):
14     def __init__(self, player = 0, direction=0, master=False):
15         self.player = player
16         self.direction = direction
17         self.master = master
18
19     def __repr__(self):
20         if self.master:
21             m = " M"
22         else:
23             m=""
24         return "P"+str(self.player)+" "+str(self.direction)+m
25
26
27
28
29
30 class Game(object):
31     def __init__(self):
32         self.currentLevel = 0
33         pygame.init()
34         #self.levels = [Level6(game=self)]
35         window_width = 840
36         window_height = 840
37         win = pygame.display.set_mode((window_width, window_height))
38         self.window = win
39         ini = time()
40         self.board = [[0]+[Piece(i%2,1) for i in range(6)]+[0]]+[[Piece(i%2,0)]+[0 for i in range(6)]+[Piece(i%2,0)] f
41         self.buttons = [[i for i in range(8)] for j in range(8)]
42         self.selected = None
43         self.active_player = 0
44         self.quit = False
45         self.menu()
46         self.play()
47         pygame.quit()
48
49     def play(self):
50         for i in range(len(self.board)):
51             for j in range(len(self.board[i])):
52                 self.buttons[i][j] = pygame.draw.rect(self.window, ((i+j)%2*255, (i+j)%2*255, (i+j)%2*255), (20+j*100,
53         self.draw()
54         while not (self.game_over() or self.quit):
55             self.control()
56
57         pygame.quit()
```