UFRN
DCA 0413
CONTROLE INTELIGENTE

Jogo da Velha

Aluno Marco Antonio Moreira Carujo

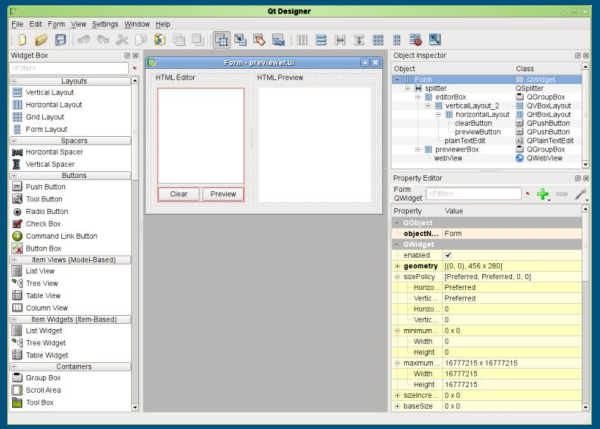
Tópicos

- Interface em QT
- Linguagem de programação Python
- Hard Rules
- Tomada de decisão com Minmax
- Apresentar o Jogo

- Referências

Interface em QT

- QT5 Designer
- file.ui (XML)
- pyuic5




```xml
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Dialog</class>
 <widget class="QDialog" name="Dialog">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>657</width>
    <height>516</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Dialog</string>
  </property>
  <widget class="QPushButton" name="B00">
   <property name="geometry">
    <rect>
     <x>230</x>
     <y>100</y>
     <width>51</width>
     <height>51</height>
    </rect>
   </property>
   <property name="font">
    <font>
     <pointsize>26</pointsize>
```

```python
# Form implementation generated from reading ui file 'Interfa
#
# Created by: PyQt5 UI code generator 5.5.1
#
# WARNING! All changes made in this file will be lost!

from PyQt5 import QtCore, QtGui, QtWidgets
from game import Game
import time
class Ui_Dialog(object):
    play_mark = True
    def setupUi(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.resize(657, 516)
        self.B00 = QtWidgets.QPushButton(Dialog)
        self.B00.setGeometry(QtCore.QRect(230, 100, 51, 51))
        font = QtGui.QFont()
        font.setPointSize(26)
        font.setBold(True)
        font.setWeight(75)
        self.B00.setFont(font)
        self.B00.setText("")
        self.B00.setObjectName("B00")
        self.B01 = QtWidgets.QPushButton(Dialog)
```

Programação em Python

- Assim como Python hoje existem inúmeras linguagens no mercado que são dinamicamente tipadas.
- Compatibilidade com QT para criação de interfaces.
- Linguagem que está em alta, com muita demanda de profissionais e ampla gama de aplicações.
- Pouca prática com a linguagem me fizeram escolher está linguagem.

Hard Rules

- Possibilidade do computador deixar de ganhar caso tome a decisão errada.
- Possibilidade do computador perder caso tome a decisão errada.
- Capacidade de que o usuário tente criar duas situações de vitória o que torna inevitável a derrota do computador.

```
6   class Game:
7       def my_time(self, table, level):
8           board_local = Board(table)
9           tabela = self.formatt_talble(table)
10
11          if(type(board_local.where_i_win()) == list and int(level) >= 1 ):
12              line, colm = board_local.where_i_win()
13              return [line, colm]
14          elif(type(board_local.where_i_lose()) == list and int(level) >= 2):
15              line, colm = board_local.where_i_lose()
16              return [line, colm]
17          elif(type(board_local.jogada_1()) == list and int(level) >= 3):
18              line, colm = board_local.jogada_1()
19              return [line, colm]
20          else:
21              auxTree = Tree(tabela,0)
22              auxTree.build_min_max_with_depth()
23              line, colm = auxTree.wich_one_is_the_best()
24              return [line, colm]
```

Tomada de decisão com Minmax

- Hierarquia de Objetos no Python com as classes: Tree,Node e Board.
- Estrutura de Árvore.

```python
class Tree:
    root = [None]
    depth = 3

    def __init__(self, table,depth):
        node = Node(table, None, None)
        self.root = node
        self.depth = depth

```

```python
class Node:
    node = [None]
    weight = None
    lineMarked = None
    colmMarked = None
    nextNode = list

    def __init__(self, table, lineMarked, colmMarked):
        auxBoard = Board(table)
        auxBoard.setTable(table)
        self.node = auxBoard
        self.lineMarked = lineMarked
        self.colmMarked = colmMarked
        self.weight = self.node.how_many_times_i_can_lose()
```

```python
class Board:
    tabela = [
                [None, None, None],
                [None, None, None],
                [None, None, None]
            ]
    size = 3

    def __init__(self, table):
        self.tabela = table
```

Tomada de decisão com Minmax

```
34
35      def how_many_times_i_can_lose(self):
36          count = 0
37          auxTable = copy.deepcopy(self.tabela[:])
38          auxTable2 = copy.deepcopy(self.tabela[:])
39          for i in range(self.size):…
45
46          for i in range(self.size):…
50
51          for i in range(self.size):…
55
56          if (auxTable[0][0] + auxTable[1][1] +…
59
60          if (auxTable[0][2] + auxTable[1][1] +…
63
64          count2 = 0;
65
66          for i in range(self.size):…
72
73          for i in range(self.size):…
77
78          for i in range(self.size):…
82
83          if (auxTable2[0][0] + auxTable2[1][1] +…
86
87          if (auxTable2[0][2] + auxTable2[1][1] +…
90
91          return count2-count
92
```

```
33      def wich_one_is_the_best(self):
34          auxNode = [ ]
35
36          Weight = -100
37          for val in self.root.nextNode:
38              if(Weight > val.weight):
39                  Weight = val.weight
40                  auxNode = val
41
42          if(self.depth == 1):
43              for x in range(len(self.root.nextNode)):
44                  Weight = -100
45                  for val in self.root.nextNode[x].nextNode:
46                      if(Weight > val.weight):
47                          Weight = self.root.nextNode[x].weight
48                          auxNode = self.root.nextNode[x]
49
50          if(self.depth == 2):
51              for x in range(len(self.root.nextNode)):
52                  for y in range(len(self.root.nextNode[x].nextNode)):
53                      Weight = -100
54                      for val in self.root.nextNode[x].nextNode[y].nextNode:
55                          if(Weight > val.weight):
56                              Weight = self.root.nextNode[x].weight
57                              auxNode = self.root.nextNode[x]
58
59          if(self.depth == 3):
60              for x in range(len(self.root.nextNode)):
61                  for y in range(len(self.root.nextNode[x].nextNode)):
62                      for z in range(len(self.root.nextNode[x].nextNode[y].nextNode)):
63                          Weight = -100
64                          for val in self.root.nextNode[x].nextNode[y].nextNode[z].nextNode:
65                              if(Weight > val.weight):
66                                  Weight = self.root.nextNode[x].weight
67                                  auxNode = self.root.nextNode[x]
68          auxNode.nextNode = None
69          # return auxNode
70
71          return [auxNode.lineMarked,auxNode.colmMarked]
```

Referências

Wikipedia, Jogo da Velha. Disponível em:<https://pt.wikipedia.org/wiki/Jogo_da_velha> Acesso em: 01 de Abril de 2018

Python, Documentação Python3. Disponível em:<https://www.python.org/> Acesso em: 25 de Março de 2018

Wikipedia, Minmax. Disponível em:<https://pt.wikipedia.org/wiki/Minimax> Acesso em: 28 de Março de 2018

Henrique Vianna, Fundamentos de IA. Disponível em:<http://henriquevianna.com/code/ia/jogo-da-velha.html>
Acesso em: 23 de Março de 2018

QT, Documentação. Disponível em:<https://www.qt.io/qt-features-libraries-apis-tools-and-ide/>
Acesso em: 25 de Março de 2018

Wikihow, 3 Formas de ganhar no Jogo da Velha. Disponível em: <https://pt.wikihow.com/Ganhar-no-Jogo-da-Velha>
Acesso em: 02 de Abril de 2018