```
# ==============================================================================
"""PENTE : implementation of the "Penté" board game for two players"""
# ==============================================================================
__author__ = "Gwendal Prat and Clément Thion"
__version__ = "0.0"
__date__ = "2021-12-01"
# ==============================================================================
from ezTK import *
# ==============================================================================
class ConfigWin(Win):
  """configuration window for the "Penté" game"""
  def __init__(self, dim=8, score=5, nameA='Player A', nameB='Player B'):
    """create and show the configuration window"""
    font = 'Arial 20 bold'
    Win.__init__(self, title="ConfigWin",op=2,grow=True,bg='#98f4f3')#config window
    # ---------------------------------------------------------------------------
    Label(self,text='CONFIGURATION',width=23,bg='#1b32c5',fg='#819b93',font=font)
    # ---------------------------------------------------------------------------
    frame = Frame(self, fold=2, flow='ES')
    #----
    Label(frame,text='Name of player A :',width=13,anchor='SW',grow=True,
          font='Arial 13 bold')
    self.nameA = Entry(frame)
    self.nameA.insert(0,'Player A')
    #----
    Label(frame,text='Name of player B :',width=13,anchor='SW',grow=True,
          font='Arial 13 bold')
    self.nameB = Entry(frame)
    self.nameB.insert(0,'Player B')
    #----
    Label(frame,text='Board Dimensions :',width=16,anchor='SW',grow=False,
          font='Arial 13 bold')
    self.dim = Scale(frame, scale=(dim, 16), flow='W', state=dim)
    #----
    Label(frame,text='Score for Victory :',width=16,anchor='SW',grow=False,
          font='Arial 13 bold')
    self.score = Scale(frame, scale=(score, 15), flow='W', state=score)
    # ---------------------------------------------------------------------------
    Button(frame,text='⚙',command=self.settings,bg='#1b32c5',fg='#819b93',
           font=font)
```

```
    Button(frame,text='?',command=self.rules,bg='#1b32c5',fg='#819b93',
           font=font)
    Button(frame,text='START',command=lambda: GameWin(self),bg='#1b32c5',
           fg='#819b93',font=font)
    #-----------------------------------------------------------------------------
    self.gchrono=IntVar(); self.pchrono=IntVar();
    self.voisinage=IntVar()
    self.timelimite=120 #time limit per player in seconds
    # -----------------------------------------------------------------------------
    self.loop()
  # -----------------------------------------------------------------------------
  def settings(self):
    """callback for the "POPUP" button"""
    settingswin = Win(self, title='SETTINGS', flow='S', op=10,bg='#98f4f3')
    Label(settingswin, text='SETTINGS',fg='#819b93',bg='#1b32c5',
          font='Arial 20 bold')
    frame = Frame(settingswin,flow='ES', fold=2)
    #----
    Label(frame, text='Add a global Chrono ?',font='Arial 13 bold',
          width=30,anchor='W')
    Checkbutton(frame, variable=self.gchrono)
    #----
    Label(frame, text='Add a player turn Chrono ?',font='Arial 13 bold',
          width=30,anchor='W')
    Checkbutton(frame, variable=self.pchrono)
    #----
    Label(frame,text='Time limit per player (seconds)',font='Arial 13 bold',
          width=30,anchor='W')
    # ---------------------
    def save_choosechrono():
      self.timelimite=self.choosechrono.state
    self.choosechrono = Scale(frame, scale=(60,600), state=120,
                              command=save_choosechrono)
    #----------------------
    Label(frame, text='Add the switch rule ?',font='Arial 13 bold',width=30,
          anchor='W')
    Checkbutton(frame, variable=self.voisinage)
    #----
    Button(settingswin, text='CLOSE', command = settingswin.exit,
           fg='#819b93',bg='#1b32c5', font='Arial 20 bold')
```

```
  # ------------------------------------------------------------------------
  def rules(self):
    popup = Win(self, title='RULES', op=10,bg='#98f4f3')
    Label(popup, text= "This game is called Pente \n When one of the two \
    players reaches the chosen victory score (set before the game starts), the \
    game is over. \n To collect points, the players can either form a 5-tokens\
    row or column, or set 2 tokens aside an opponent's token. \n Warning ! You \
    cannot set a token on a cell if there's already a token, or if the cell is \
    next to a token set the turn before.")
    Button(popup,text='UNDERSTOOD',command=popup.exit,fg='#819b93',bg='#1b32c5',
           font='Arial 20 bold')
    popup.wait()
#=============================================================================
class GameWin(Win):
  """game window for the "Penté" game"""
  def __init__(self, config):
    """create and show the game window, according to config parameters"""
    # ----GETTING SETTINGS-------------------------------------------------
    # ----Subsettings -----------------------------------------------------
    self.choosechrono = config.timelimite
    self.gchrono = True  #Global chrono condition
    if config.gchrono.get() == 0: self.gchrono = False
    self.pchrono = True  #Player chrono condition
    if config.pchrono.get() == 0: self.pchrono = False
    self.voisinage = True  #True by default
    if config.voisinage.get() == 0: self.voisinage = False
    # ------------------------------------------------------------------------
    self.dim = config.dim.state
    self.game = Game(self.dim)  # create kernel class and store it as attribute
    self.NameA = config.nameA.state  #'NAME A'
    self.NameB = config.nameB.state  #'NAME B'
    self.score = config.score.state
    # ------------------------------------------------------------------------
    config.exit()
    self.show()
  # ------------------------------------------------------------------------
  def on_click(self, widget, code, mods):
    """callback function for all mouse click events"""
    #assert to click on the board and nowhere else
    if widget.master != self.frame or widget.index is None: return
```

```
    # ------------------------------------------------------------------------
    if code != 'LMB': return
    else:  #when LeftMouseButton
      if widget.state != 0: return  #can only play on a black cell
      #--
      self.game.playerID=2-(1 + self.game.playerID) % 2#update player id in(1, 2)
      row, col = widget.index  #coordinates of clicked Label
      self.game.history.append((row, col))
      row0, col0 = self.game.history[-2] #get last move before current clicl
      #---- rules applications : Game calls ==> updates of state matrice MStates
      self.game(row, col,self.game.playerID)  #update states matrice for player
      if self.voisinage: #if neighborhood rule is active in settings
          self.game.switch(row0, col0, True)  #erase last switch
          self.game.switch(row, col, False)  #new current switch
      self.game.align(row, col, self.game.playerID)
      self.game.capture(row, col, self.game.playerID)
      self.victory()
      #-- convert Game.L state into graphical animation
      self.tour.state = (self.tour.state + 1) % 2  #equivalent to do +=1
      for cell in self.game.Changes:  #update the grid
              self.frame[cell[0]][cell[1]].state = self.game(
                  cell[0], cell[1])
      #-- update score on display
      self.A['text'] = f'{self.NameA} \n {self.game.score[0]}'
      self.B['text'] = f'{self.NameB} \n {self.game.score[1]}'
      self.game.Changes = []
# ------------------------------------------------------------------------
def show(self):
    """show current game board by setting state defined for each grid cell"""
    # ------------------------------------------------------------------------
    Win.__init__(self,title="Pente",op=2,fold=1,flow='ES',click=self.on_click,
                 bg='#98f4f3',
                 grow=False)  #creates window
    font2 = 'Arial 20 bold'
    images = tuple(Image(file=f"{id}.gif")for id in range(4))  #import image
    # ------------------------------------------------------------------------
    if self.gchrono == True:
      Label(self,text = 'Global Time :', font="Arial 16 bold underline")
      self.globalchrono = Label(self, text=0, font="Arial 16 bold")
```

```
    time = Frame(self,fold=2,flow='ES')
    if self.pchrono == True :
      Label(time,text = f"{self.NameA}'s time left", font="Arial 16 bold")
      Label(time,text = f"{self.NameB}'s time left", font="Arial 16 bold")
      self.chrono1=Label(time,text=self.choosechrono,font='Arial 16 bold',
                         width=3)
      self.chrono2=Label(time,text=self.choosechrono,font='Arial 16 bold',
                         width=3)

    frameStat = Frame(self, flow='ES')
    self.A = Label(frameStat,font=font2,fg='blue',border=2,width=10,
          text=(f'{self.NameA}\n{self.game.score[0]}'))#Player A informations
    self.tour = Label(frameStat,font='Arial 35 bold',width=2,
          text=('A','B'),bg='Black',fg=('#6069f5','#50db20'))#Current player turn
    self.B = Label(frameStat, font=font2,fg='green',border=2,width=10,
          text=(f'{self.NameB}\n{self.game.score[1]}'))#Player B informations
    # ------------------------------------------------------------------------
    width, height = self.winfo_screenwidth()-64, self.winfo_screenheight()-64
    step = min(width/self.dim, height/self.dim)
    self.frame = Frame(self, fold=self.dim, flow='ES',
                       width=step*self.dim, height=step*self.dim)#grid container
    for n in range(self.dim * self.dim):  #Creates the grid
      grid = Label(self.frame, image=images)
    # ------------------------------------------------------------------------
    self.after(2000,self.tick);self.loop()
# ----------------------------------------------------------------------------
def victory(self):
    """check victory condition and play victory animation"""
    if self.game.score[0] >= self.score :
      self.winner = self.NameA
      self.game.over=True
    elif self.game.score[1] >= self.score:
      self.winner = self.NameB
      self.game.over=True

    if self.game.over:
      for r in range(self.dim):
        for c in range(self.dim):
          self.frame[r][c].state = 3
      # --Win--
```

```
      popup = Win(self, title='VICTORY', op=10,bg='#98f4f3')
      Label(popup, text="Game Over",font = 'Arial 30 bold underline')
      if self.gchrono == True :
        Label(popup, text= f"The winner is {self.winner}. This game lasted \
{self.globalchrono['text']} seconds.")
      else:
        Label(popup, text= f"The winner is {self.winner}.")
      # --Frame--
      frame = Frame(popup, fold=2)
      Label(frame, text = f"{self.NameA}'s results :",font='Arial 15 underline')
      Label(frame, text = f"{self.NameB}'s results :",font='Arial 15 underline')
      Label(frame, text = f"{self.game.score[0]} points")
      Label(frame, text = f"{self.game.score[1]} points")
      if self.pchrono == True :
        Label(frame, text = f"{self.chrono1['text']} second(s) remaining")
        Label(frame, text = f"{self.chrono2['text']} second(s) remaining")
      # --//--
      Button(popup,text='NEW GAME',command=self.new_game,bg='#1b32c5',
             fg='#819b93',
             font='Arial 20 bold')
      popup.wait()
  # ------------------------------------------------------------------------
  def new_game(self):
    """New game launcher"""
    self.exit();ConfigWin()
  # ------------------------------------------------------------------------
  def tick(self):
    """Manage chrono"""
    self.victory()  #stop the chrono if there's a winner
    #----Player chrono only
    if self.pchrono==True :
      if self.chrono1['text']==0 :
        self.game.over=True
        self.winner = self.NameB
        self.victory()
        return
      if self.chrono2['text']==0:
        self.game.over=True
        self.winner=self.NameA
        self.victory()
```

```
            return
        #----Global chrono only
        if self.gchrono == True and self.pchrono == False:
          self.globalchrono['text']+=1
          self.after(1000,self.tick)
        elif self.gchrono == False and self.pchrono == True:
          if self.tour.state == 0:
            self.chrono1['text'] = self.chrono1['text'] -1
            if self.chrono1['text'] == 20:self.chrono1['fg'] = 'red'
            self.after(1000, self.tick)
          else:
            self.chrono2['text'] = self.chrono2['text'] -1
            if self.chrono2['text'] == 20: self.chrono2['fg'] = 'red'
            self.after(1000, self.tick)
        #----Both chrono
        elif self.gchrono==True and self.pchrono==True:
          self.globalchrono['text']+=1
          if self.tour.state == 0:
            self.chrono1['text'] = self.chrono1['text'] -1
            if self.chrono1['text'] == 20:self.chrono1['fg'] = 'red'
            self.after(1000, self.tick)
          else:
            self.chrono2['text'] = self.chrono2['text'] -1
            if self.chrono2['text'] == 20: self.chrono2['fg'] = 'red'
            self.after(1000, self.tick)
    # ============================================================================
    class Game(object):
      """kernel class for the "Penté" game"""
      def __init__(self, dim=8):
        """create and initialize the grid data structure"""
        self.over = False  #game over indicator
        self.dim = dim
        self.history = [(0, 0)]#storages every players moves from start to game over
        self.MState=[dim * [0].copy()for _ in range(dim)]#bijection with grid states
        self.Changes = []  #list of points modified by current move
        self.score = [0, 0]  #storage of players score
        self.playerID = 2  #last player who played, 2 by default for the first move
      # ------------------------------------------------------------------------
      def __call__(self, row, col, state=None):
        """get or set state for provided grid cell. Control cell validity (in grid)"""
```

```
  if not 0 <= row < self.dim or not 0 <= col < self.dim: #out of grid
      return
  if state == None:
      return self.MState[row][col]  #return cell state
  else:  #change cell state
    assert isinstance(state, int), "state must be an integer"
    self.MState[row][col] = state % 4
    self.Changes.append((row, col))
# ----------------------------------------------------------------------------
def switch(self, row, col, valid=True):
  """switch valid/invalid state for neighborhood of provided grid cell"""
  #----Neighborhood
  neighborhood=[(1, 1),(1, 0),(1, -1),(0, 1),(0, -1),(-1, 1),(-1, 0),(-1, -1)]
  for x in neighborhood:
    xrow, xcol = row + x[0], col + x[1]  # x,y neighbor coordinates
    if not 0 <= xrow < self.dim or not 0 <= xcol < self.dim:
      continue  #if out of grid
    elif not valid and self.MState[xrow][xcol] == 0:  #if black cell
      self(xrow, xcol, 3)  #--> grey
    elif valid and self.MState[xrow][xcol] == 3:  #if grey cell
      self(xrow, xcol, 0)  #--> black
# ----------------------------------------------------------------------------
def align(self, row, col, playerID):
  """check if provided move creates align config and return score update"""
  neighborhood = [[0, 1], [1, 0], [-1, 1], [1, 1]]
  for x in neighborhood:
      line=[self(row+(-5+k)*x[0], col+(-5+k)*x[1]) for k in range(1,10)]
      linestr=''.join([str(l) for l in line])
      if 5*str(playerID) in linestr:
          self.score[playerID - 1] += 5 #update score
# ----------------------------------------------------------------------------
def capture(self, row, col, playerID):
  """check if provided move creates capture config and return score update"""
  adversaryID = 2-(1+playerID)%2
  neighborhood=[(0, 1),(-1, 1),(-1, 0),(-1, -1),(0, -1),(1, -1),(1, 0),(1, 1)]
  for x in neighborhood:
    line=[self(row+k*x[0], col+k*x[1]) for k in range(4)]
    if line == [playerID, adversaryID, adversaryID, playerID]:
        self.score[playerID - 1] += 1 #update score
        for k in range(1,3):self(row+k*x[0],col+k*x[1],0)#delete opponent
```

```
                                                                    #captured token
# ================================================================================
if __name__ == "__main__":
  ConfigWin()
# ================================================================================
```