Въведение:

Проекта който направих е познатата ни на всички игра, а именно Тетрис.  
Имаме игрално поле от горния край, на което падат различни по цвят и форма елементи. Целта ни е така да ги завъртим(това става с помощта на клавиатурата), че като стигнат дъното на игралното поле да заемат място до останалите и когато се напълни цял ред (без останали празни места) той се унищожава и играчът получава една точка. Чрез определен брой натрупани точки се преминава на по – горно ниво и елементите започват да падат по – бързо. През цялото време играчът вижда кой е следващият елемент, който се очаква да се появи на игралното поле. Играта завършва след като някой от елементите стигне до долния край на игралното поле. За удобство на играча, той може да използва пауза по всяко време на играта. А за по – приятна атмосфера по време на игра има и музика. Ако искате да играете тетрис заедно с ваш приятел може да го направите като изберете от менюто “Two Players” и ще се появят две игрални полета. Можете да играете заедно стига да сте на един и същ компютър.

Бележка: За да играете играта трябва да имате инсталиран python и pygame

**Описание на приложените алгоритми**

Използвам **random** за получаване на елементите на случаен принцип,  
**timе** - за връщане към сегашното време (когато е паднал елемента вече time позволява времето един вид да се update-не за да може следващия елемент да падне), а също и за придвижването на елементите в различните посоки,  
**sys** съдържа информация за константите, функциите, и методите които се използват.  
Чрез **pygame** рисувам различните обекти, правя update на играта, когато е необходимо, също го използвам и за зареждането и спирането на музикалния фон към играта, както и за настройването на клавишите, с които да бъде управлявана играта и др.

Описание на програмния код

class drowBox:

*def drawBox(boxx, boxy, color, pixelx=None, pixely=None):*функцията рисува отделното квадратче, всяка фигура се състои от четири такива,  
рисува ги на (x,y) кординати на дъската, а ако pixelx и pixely са определени ги рисува там, където са определени вече да бъдат( това се използва за рисуване на следващия елемент)

class colors

в него се съхраняват цветовете, както и се проверява дали всеки цвят има съответен по – светъл такъв

class\_aboutPieces

*def drawNextPiece(piece):*Отговаря за рисуване на следващия елемент отстрани на игралното поле. Това става чрез извикване на функцията drawPiece и задаване на координатите, където искаме да се намира фигурата

*def drawPiece(piece, pixelx=None, pixely=None):*отговаря за рисуване на екрана на елемента, който трябва да падне в момента   
Чрез вложен цикъл обхождаме координатите на фигурата и изрисуваме всяка кутийка, от която е съставен елементът. Всеки елемент е с височина и дължина 5 на 5 като тези които са точки в шаблона се пренебрегват с помощта на if и не се изобразяват.

*def getNewPiece():*

Прави се речник като в него се задава как да се определят точно формата, как да бъде завъртян елемента, координатите (х, у), на които да се появи на игралното поле както и коя от възможните форми да бъде. След което return-ваме речника и така се появява новата фигура на игралното поле.

class Game

*def \_\_init\_\_(self, size, title, fps) е конструктор*

*def events(self)*тук се съхраняват част от събитията, които могат да настъпят в играта

*def runGame*()   
тук е основния код на играта, също функцията отговаря и за управлението на самите елементи от клавиатурата.  
С помощта на множество if-ове задаваме кой клавиш от клавиатурата като бъде натиснат, какво да се случи. Чрез извикване на различни функции запълваме игралното поле. Пример: aboutBoard\_.drawBoard(board) – рисува самото игрално поле

aboutBoard\_.drawStatus(score, level) – рисува точките и нивото

aboutPieces.drawNextPiece(nextPiece) – отговаря за появяването на следващия елемент отстрани на игралното поле

def runGame2()  
Отговаря за същите неща като defrunGame, но тук те са за двамата играчи

*def terminate()*чрез тази функция играта се изключва с помощна на pygame.guit() и sys.exit()

class aboutBoard\_:

*def drawBoard(board)*С помощта на pygame с тази функция рисуваме дъската около игралното поле и запълваме фона. Както и обхождаме с помощта на вложен цикъл кутийките по дъската за ги нарисуваме.

*def getBlankBoard()*Имаме цикъл който обхожда хоризонтално игралното поле и на всеки координат, на който се намираме слагаме BLANK по височината му, тоест изчистваме игралното поле от фигурите, за да получим ново такова.

class Sound(object)

Класът отговаря за звуковия фон на играта, а именно коя песен да започне, кога да започне, кога да спре, или да е на пауза. Това става с помощта на следните функции:

def play(self)

Има цикъл (while), който докато има стойност True се зарежда една от двете песни, между които има избор, там се извиква и runGame(), което слага началото на играта. Приключването на функцията ознавачава, че играта е свършила, затова музиката се спира и след извикване на функцията showTextScreen('Game Over')на екрана се появява съобщение за край на играта.

*def pause(self)*

С помощта на pygame музиката спира и чрез извикване на aboutBoard\_.showTextScreen(Paused) на екрана се появява текст Pause, след като е натиснат клавиш, музиката се стартира отново.

*def stop(self)*

С помощта на pygame музиката се спира

class Sound2(object)

Същия като class Sound(object) с разликата че се извиква runGame2(), а не runGame()

class Lines:  
*def addToBoard(board, piece)*Падналия елемент да се счита като част от игралното поле вече и по - точно за дъно на игралното поле чрез вложен цикъл за обхождане на големината на фигурата, която е паднала.

*def isOnBoard(x, y)*Проверка дали кодринатите (x, y) са от игралното поле

*def isValidPosition(board, piece, adjX=0, adjY=0)*  
функцията връща True, ако координатите на елемента са от игралното поле и елемента не се е сблъскал с някоя от стените на игралното поле. Става с вложен цикъл чрез обхождане координатите на елемента.

*def isCompleteLine(board, y)*проверка дали има изцяло запълнен ред  
Функцията връща True, ако има такъв и False, ако няма .Става чрез обхождане на ширината на игралното. поле. След което if, където се проверява дали има BLANK и ако няма връща True, а ако има False

*def removeCompleteLines(board)*Премахване на изцяло запълнените редове  
Става чрез цикъл, който се изпълнява докато y>0. Когато това е така и има запълненен ред преместаме y с едно надолу( т.е. y=y-1), след което поставяме BLANK ред най отгоре на игралното поле

Други функции

*def makeTextObjs(text, font, color)*Отговаря за правенето на текст, който да се появява на екрана

*def calculateLevelAndFallFreq(score)*Спрямо точките, връща нивото, на което се намира играча, както и това, колко време да мине преди елемента да се предвижи с едно квадратче надолу по игрално поле. Тоест като се извика нивото се увеличава с 1, и фигурата се забързва.

*def start\_main\_menu():*

С извикването на функцията се стартира играта (т.е върши работата на main-функция)

Извиква функцията showTextScreen('Tetris'), за да се появи голям надпис Тетрис на игралното поле, с натискане на клавиш се стартира менюто.

*def start\_single\_player\_level\_menu():*

Използва се за извикване на Sound.play(), което стартира играта

def start\_multiplayer\_level\_menu():

Използва се за извикване на Sound.play2(), което стартира играта

*def handle\_menu\_event(menu):*

Чрез for и if се задава, кой клавиш ако бъде натиснат, какво да се случи (подобно на това в runGame()) Ако играта не бъде изключена да се зареди start\_main\_menu, което ни изкарва опции, между, които да се избира каквото желае потребителя. След като бъде заредено менюто се проверява кой бутон е натиснат, за да може да се зареди именно тази опция, също се проверява и за натиснат бутон на мишката.

class MenuOption (pygame.font.Font):

*def set\_position(self, x, y):*

Позиционира менюто

*def highlight(self, color=colors.GREEN):*

Ако бъде маркирана опция от менюто, тя да се изобразява в друг цвят. Става с помощта на

self.font\_color = color

self.label = self.render(self.text, 1, self.font\_color)

self.is\_selected = True

и така ако е извикана тази функция се приема че опцията е избрана

*def unhighlight(self):*

По начало опциите да са с определен цвят, става подобно на предна функция.

*def check\_for\_mouse\_selection(self, mouse\_pos):*

Проверка дали е избрана опция или не .  
Ако условието (if self.rect.collidepoint(mouse\_pos)) е изпълнено се извиква функцията self.highlight(), а ако не е self.unhighlight(),