”python tkinter로 테트리스 만들기”

부산대학교 정보컴퓨터공학부
201424465 배준혁(dlfltltm2@gmail.co.kr)

2019년 10월 19일

1 개요

파이썬과 tkinter 모듈을 사용하여 GUI(Graphic Use Interface)를 지원하는 게임을 개발, 기본적인 프로그래밍 실력을 향상하고자 함. 또한 게임을 만들면서 배운 내용을 문서화 하여 문서작성 능력을 키우고 새로운 지식을 공유하려함.

2 python tkinter

2.1 tkinter

tkinter는 파이썬 표준 GUI 모듈이다. tkinter는 타 GUI 프레임워크나 툴킷에 비해 지원되는 위젯들이 부족하고 UI도 투박하지만, 파이썬 설치시 기본적으로 내장되어 있는 파이썬 표준 라이브러리이기 때문에 간단한 GUI 프로그램을 만들 때 유용하다.

2.2 간단한 tkinter 프로그램

```
import tkinter
root = tkinter.Tk()
root.mainloop()
```

코드1: 간단한 tkinter 프로그램

위와 같이 tkinter는 파이썬 기본 모듈이므로 설치할 필요없이 임포트하여 사용하면 된다. mainloop()는 계속 동작하면서 키보드나 마우스등의 입력 이벤트를 메세지로 받아서 핸들러에 전달하는 역활을 한다.

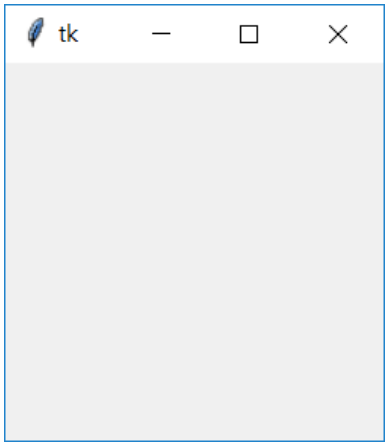

그림 1: tkinter window

코드1(2.2) 로 생성된 아무 기능도 없는 tkinter window의 모습.

버튼이나 입력같은 다른 기능이 필요할 경우 mainloop()가 실행되기 전에 Label, Entry, Button과 같은 클래스를 만들어준뒤 클래스 함수인 pack()을 통하여 윈도우에 넣어서 사용자에게 보여줄 수 있다. 이렇게 생성된 각 클래스들은 이벤트와 핸들러를 바인딩하여 원하는 기능을 간단하게 구현할 수 있다.

3 테트리스

1984년 소련과학원에서 근무하던 컴퓨터 프로그래머 '알렉세이 파지노프'가 개발한 테트리스는 전통 퍼즐게임인 '펜토미노'를 개량해서 만든 게임이다. 게임은 단순하다. 4개의 사각형 조합으로 낙하하는 7가지 모양의 테트리미노 블록을 조립해서 한 줄을 꽉 채우면 한 줄이 사라진다.

그림 2: ATARI Tetris

오락실에서 한번쯤은 봤을 아타리의 아케이드판 테트리스. 테트리스는 수천개의 아류작이 있는 게임으로도 유명하다.

테트리스는 또한 초보 프로그래머가 도전하기 좋은 과제이기도 하다. GUI 프로그래밍과 키보드를 이용한 입력, 블록간 충돌을 구현하기 위한 논리적 연산, 점수 계산과 파일 입출력등 다양한 기능들을 만들면서 고민을 하면 해당 프로그래밍 언어와 여러 프로그래밍 개념에 대해 이해할 수 있다.

이번 프로젝트에서는 파이썬과 tkinter 모듈을 사용하여 테트리스를 구현, 프로그래밍 연습과 재미의 두마리 토끼를 모두 잡으려 노력했다.

4 파이썬 테트리스의 구현

이제 테트리스를 파이썬 으로 구현해보자.

4.1 테트로미노의 구성

테트리스는 4종류의 테트로미노를 이용한 게임이다. 가장 먼저 게임에서 사용할 테트로미노를 구성해보자.

```
Z = [['.....',
      '.....',
      '.00..',
      '..00.',
      '.....'],
     ['.....',
      '..0..',
      '.00..',
      '.0...',
      '.....']]
```

코드2: 파이썬 리스트로 구현한 Z블록

파이썬 리스트를 사용하여 각 블록을 구현하기로 하였다. 이때 회전을 통해 만들어진 모양은 같은 리스트에 포함시켜 인덱스를 통하여 간단하게 회전을 구현할 수 있도록 하였다. 다른 블록들도 마찬가지 방법으로 리스트를 통하여 구현되었다.

4.2 Class GameWindow

```
class GameWindow:
    started = False
    timer = None
    totalTime = 0
    ...
    def __init__(self):
        self.root = Tk()
        self.root.title('pyTetris')
```

```
        self.mainFrame = Frame(self.root, bg='#FFF', width=350, height=
                                            600)
        self.gameFrame = Canvas(self.mainFrame, bg='white', width=350,
                                            height=600)
        self.subFrame = Frame(self.mainFrame, bg='gray', width=150,
                                            height=600)

        startbuttonWrapper = Frame(self.subFrame)
        startbuttonWrapper.grid(row=5, column=0, pady= 10)
        self.startButton = Button(startbuttonWrapper, text="start",
                                            command=self.gameStart,
                                            width=15)
        self.startButton.grid()

        ...
        self.root.mainloop()

    def gameStart(self):
        ...
    def gameLoop(self):
        ...
    def keyPressed(self):
        ...
```

코드3: tkinter window를 포함한 GameWindow 클래스

GameWindow 클래스는 게임을 구동하는 클래스로 게임의 상태를 유지하기 위한 여러 변수와 tkinter window를 가지고 있다. 게임에서 사용되는 로직에 관한 모든 함수들은 GameWindow의 내부 함수로 구현되어 있다.

GameWindow 클래스는 실제 게임 화면을 보여주는 canvas와 다음 블록의 모양, 경과시간, 점수 등이 출력되는 여러 frame들을 가지고 있다. 프로그램이 실행되어 클래스가 생성되면 초기화 함수 ___init___()을 통해 frame으로 구성된 내부 UI를 설정하고 게임시작,종료 버튼의 클릭이벤트에 대하여 각각 gameStart(), gameOver()를 바인딩한다.

모든 설정이 끝나면 mainloop()가 실행된다. 이후 게임 시작 버튼을 누를경우 바인드된 gameStart() 함수가 실행되며 게임 루프를 통해 게임이 진행된다.

4.3 게임 루프 구현

```
def gameLoop(self):
    if not self.started:
        self.timer.cancel()
        self.root.quit()
    else:
        # print total time
        self.totalTime = self.totalTime + 1
        timestr = "{:02d}:{:02d}".format(self.totalTime // 60, self.
                                            totalTime % 60)
        self.timerFrame.config(text=timestr)

        self.timer = threading.Timer(1, self.gameLoop)
        self.timer.start()
        if self.check_valid():
            self.blockDown()
        self.drawBlcok()

def drawBlcok(self):
    shape = self.currentBlock.shape
    self.gameFrame.delete('current')
    block = self.currentBlock.getCoordinates()
    for cord in block:
        row = cord['r']
        col = cord['c']
        start_x = (col) * 30
        start_y = (row) * 30
        end_x = start_x + 30
        end_y = start_y + 30
        self.gameFrame.create_rectangle(start_x, start_y, end_x, end_y,
                                            fill=shape_colors[shape],
                                        tag='current')
```

코드4: 게임의 핵심 loop 구현

게임이 시작되면 시간을 1씩 증가시키고 UI에 보여질 경과시간을 문자열로 만들어 주었다. 이 문자열은 GameWindow 내부의 timerFrame를 통해 UI에 출력된다.

또한 파이썬의 threading.Timer를 사용하여 게임프로그램의 메인 스레드가 아닌 추가 스레드 'timer'가 게임 로직을 처리하도록 하였다. 타이머는 매 1초마다 작동하여 game.loop를 다시 호출한다. 이때 다음 블록의 위치가 이동가능하다면 블록을 아래로 이동시킨다.

블록은 각 모양별로 가지게되는 좌표와 실제 블럭의 좌표를 더하여 최종 좌표를 얻게된다. drawBlock()은 gameFrame에 계산된 좌표를통해 사각형을 그린다.

4.4 핵심 규칙, 충돌 구현

```
def check_valid(self):
    block = self.currentBlock.getCoordinates()
    for cord in block:
    r = cord['r']
    c = cord['c']
    if r > 19 or r < 0 or c > 8 or c < 0 :
        return False
    return True

def check_collision(self):
    block = self.currentBlock.getCoordinates()
    for cord in block:
        row = cord['r']
        col = cord['c']
        if col < 0 or col > 8:
            return True
        if row == 20:
            return True
        if self.gamemap[row][col] != 7:
            if row == 1:
                self.gameOver()
            return True
        return False
```

코드5: 게임의 핵심규칙 구현

각 함수는 현재 블록이 차지하는 좌표 4개를 getCoordinates()를 통해 획득한다. 이후 block 내부의 좌표가 다른 블록이나 벽(경계선)에 충돌하는지를 확인하게 된다. check_valid() 함수는 블록의 충돌을 확인하여 블록이 아래로 내려갈 수 있을경우 True를 반환하는 함수이며 check_collision()함수는 블록이 땅에 도착하거나 다른 블록과 충돌할시 True를 리턴한다. 이 두 함수가 키보드입력이나 타이머를 통해 호출되어 이동이 가능한지 불가능한지를 확인하는데 사용된다.

4.5 기타 기능

이외에도 한줄을 채웠을시 점수를 주고 라인을 지우는 check_line_clear(), 블록을 교체할 수 있게해주는 홀드 기능을 구현하였다.

5 구현 결과

5.1 시작화면

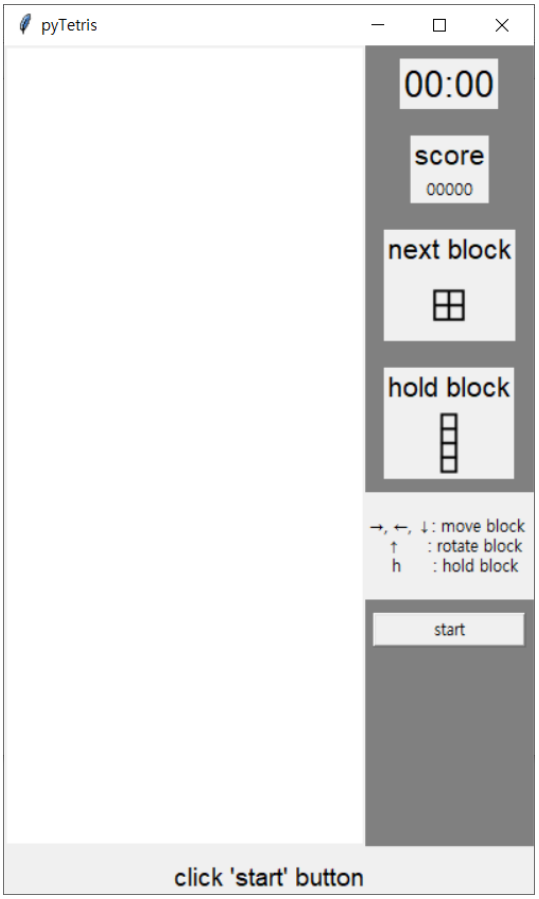

그림 3: 초기 화면

게임이 시작되면 보이는 초기화면이다. GameWindow 클래스가 생성되고 mainloop가 실행되고있다.

게임을 시작하면 그림 3과 같은 모습을 확인 할 수 있다. GameWindow의 init이 종료된 상태이며 start 버튼을 누르면 게임이 시작된다.

5.2 게임진행

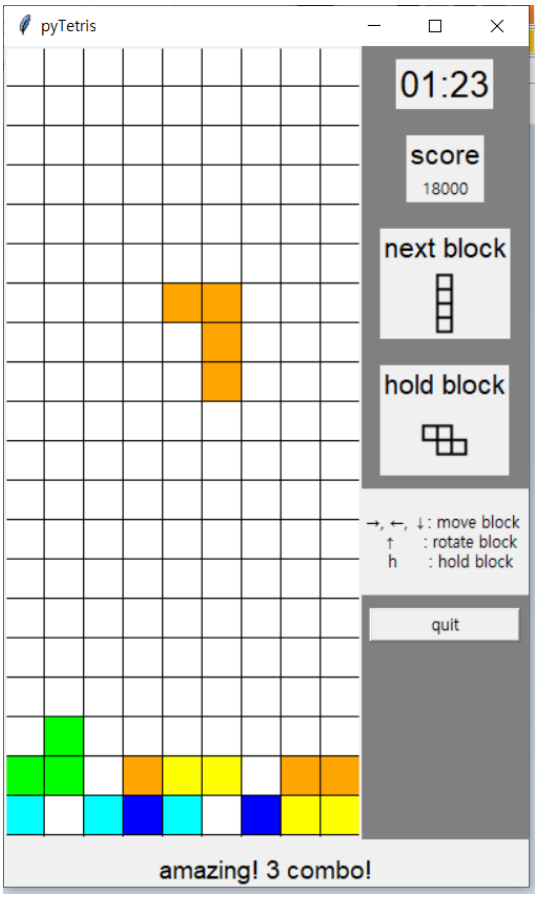

그림 4: 게임 진행

게임이 실행화면. 다음블록과 점수 시간등이 표시되는것을 확인 할 수 있다.

그림4는 게임이 진행중인 모습을 보여준이다. 경과시간, 점수, 다음블록, 홀드(저장중인)블록 등을 GUI로 확인 가능하다. 본문에선 소개되지 않은 하단 메세지 기능도 확인할 수 있다.

게임시작시 키보드 입력에관한 이벤트가 blockMove(), blockRotate()에 바인드 되기에 키보드를 통해 게임을 플레이할 수 있다.

5.3 게임종료

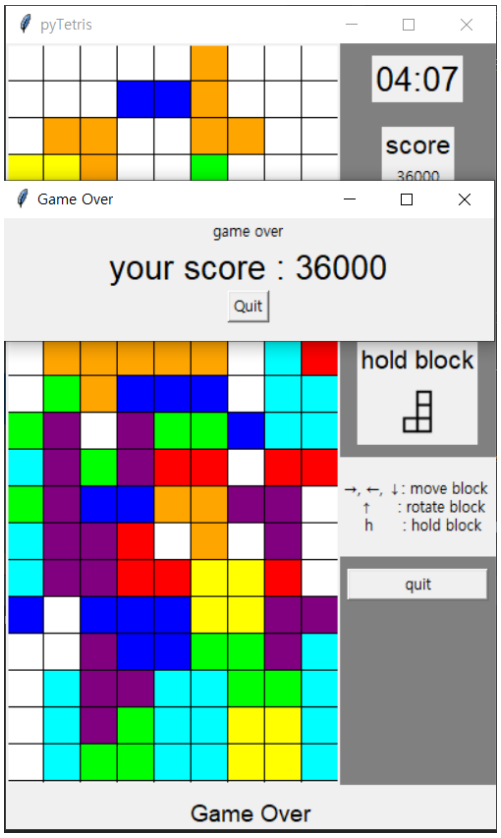

그림 5: 게임 종료

게임오버시 자신의 점수를 확인할 수 있으며 팝업창의 quit을 누르면 게임이 종료된다.

테트로미노가 꼭대기 까지 쌓이거나 직접 quit 버튼을 누를경우 게임이 종료되게 된다. 팝업창을 이용하여 자신의 점수를 확인 할 수 있다.

6 결론

파이썬을 통해 테트리스를 만들어보며 GUI 모듈인 tkinter를 사용해 보았다. 작업물을 직접 확인해보며 코딩을 하면서 논리적으로 어떤 계산을 수행해야 기능을 구현 할 수 있는지 생각하며 코딩하였다. 또한 스레드를 이용하여 메인 스레드와는 별개로 동작하는 타이머 스레드를 구현하여 입력에 즉시 반응하도록 프로그램을 짜는법을 배웠다.

`https://github.com/BaeJuneHyuck/PyTetris`에서 해당 게임의 코드를 직접 확인 할 수있다.