```
import pygame
import time
pygame.font.init()


class Grid:
    board = [
        [7, 8, 0, 4, 0, 0, 1, 2, 0],
        [6, 0, 0, 0, 7, 5, 0, 0, 9],
        [0, 0, 0, 6, 0, 1, 0, 7, 8],
        [0, 0, 7, 0, 4, 0, 2, 6, 0],
        [0, 0, 1, 0, 5, 0, 9, 3, 0],
        [9, 0, 4, 0, 6, 0, 0, 0, 5],
        [0, 7, 0, 3, 0, 0, 0, 1, 2],
        [1, 2, 0, 0, 0, 7, 4, 0, 0],
        [0, 4, 9, 2, 0, 6, 0, 0, 7]
    ]

    def __init__(self, rows, cols, width, height, win):
        self.rows = rows
        self.cols = cols
        self.cubes = [[Cube(self.board[i][j], i, j, width, height) for j
in range(cols)] for i in range(rows)]
        self.width = width
        self.height = height
        self.model = None
        self.update_model()
        self.selected = None
        self.win = win

    def update_model(self):
        self.model = [[self.cubes[i][j].value for j in range(self.cols)]
for i in range(self.rows)]

    def place(self, val):
        row, col = self.selected
        if self.cubes[row][col].value == 0:
            self.cubes[row][col].set(val)
            self.update_model()

            if valid(self.model, val, (row,col)) and self.solve():
                return True
            else:
                self.cubes[row][col].set(0)
                self.cubes[row][col].set_temp(0)
                self.update_model()
                return False

    def sketch(self, val):
        row, col = self.selected
        self.cubes[row][col].set_temp(val)

    def draw(self):
        gap = self.width / 9
        for i in range(self.rows+1):
            if i % 3 == 0 and i != 0:
                thick = 4
            else:
```

```
                thick = 1
            pygame.draw.line(self.win, (0,0,0), (0, i*gap), (self.width,
i*gap), thick)
            pygame.draw.line(self.win, (0, 0, 0), (i * gap, 0), (i * gap,
self.height), thick)

        for i in range(self.rows):
            for j in range(self.cols):
                self.cubes[i][j].draw(self.win)

    def select(self, row, col):
        for i in range(self.rows):
            for j in range(self.cols):
                self.cubes[i][j].selected = False

        self.cubes[row][col].selected = True
        self.selected = (row, col)

    def clear(self):
        row, col = self.selected
        if self.cubes[row][col].value == 0:
            self.cubes[row][col].set_temp(0)

    def click(self, pos):
        if pos[0] < self.width and pos[1] < self.height:
            gap = self.width / 9
            x = pos[0] // gap
            y = pos[1] // gap
            return (int(y),int(x))
        else:
            return None

    def is_finished(self):
        for i in range(self.rows):
            for j in range(self.cols):
                if self.cubes[i][j].value == 0:
                    return False
        return True

    def solve(self):
        find = find_empty(self.model)
        if not find:
            return True
        else:
            row, col = find

        for i in range(1, 10):
            if valid(self.model, i, (row, col)):
                self.model[row][col] = i

                if self.solve():
                    return True

                self.model[row][col] = 0

        return False

    def solve_gui(self):
```

```
        self.update_model()
        find = find_empty(self.model)
        if not find:
            return True
        else:
            row, col = find

        for i in range(1, 10):
            if valid(self.model, i, (row, col)):
                self.model[row][col] = i
                self.cubes[row][col].set(i)
                self.cubes[row][col].draw_change(self.win, True)
                self.update_model()
                pygame.display.update()
                pygame.time.delay(100)

                if self.solve_gui():
                    return True

                self.model[row][col] = 0
                self.cubes[row][col].set(0)
                self.update_model()
                self.cubes[row][col].draw_change(self.win, False)
                pygame.display.update()
                pygame.time.delay(100)

        return False


class Cube:
    rows = 9
    cols = 9

    def __init__(self, value, row, col, width, height):
        self.value = value
        self.temp = 0
        self.row = row
        self.col = col
        self.width = width
        self.height = height
        self.selected = False

    def draw(self, win):
        fnt = pygame.font.SysFont("comicsans", 40)

        gap = self.width / 9
        x = self.col * gap
        y = self.row * gap

        if self.temp != 0 and self.value == 0:
            text = fnt.render(str(self.temp), 1, (128,128,128))
            win.blit(text, (x+5, y+5))
        elif not(self.value == 0):
            text = fnt.render(str(self.value), 1, (0, 0, 0))
            win.blit(text, (x + (gap/2 - text.get_width()/2), y + (gap/2
- text.get_height()/2)))

        if self.selected:
```

```
            pygame.draw.rect(win, (255,0,0), (x,y, gap ,gap), 3)

    def draw_change(self, win, g=True):
        fnt = pygame.font.SysFont("comicsans", 40)

        gap = self.width / 9
        x = self.col * gap
        y = self.row * gap

        pygame.draw.rect(win, (255, 255, 255), (x, y, gap, gap), 0)

        text = fnt.render(str(self.value), 1, (0, 0, 0))
        win.blit(text, (x + (gap / 2 - text.get_width() / 2), y + (gap /
2 - text.get_height() / 2)))
        if g:
            pygame.draw.rect(win, (0, 255, 0), (x, y, gap, gap), 3)
        else:
            pygame.draw.rect(win, (255, 0, 0), (x, y, gap, gap), 3)

    def set(self, val):
        self.value = val

    def set_temp(self, val):
        self.temp = val


def find_empty(bo):
    for i in range(len(bo)):
        for j in range(len(bo[0])):
            if bo[i][j] == 0:
                return (i, j)

    return None


def valid(bo, num, pos):
    for i in range(len(bo[0])):
        if bo[pos[0]][i] == num and pos[1] != i:
            return False

    for i in range(len(bo)):
        if bo[i][pos[1]] == num and pos[0] != i:
            return False

    box_x = pos[1] // 3
    box_y = pos[0] // 3

    for i in range(box_y*3, box_y*3 + 3):
        for j in range(box_x * 3, box_x*3 + 3):
            if bo[i][j] == num and (i,j) != pos:
                return False

    return True

def redraw_window(win, board, time, strikes):
    win.fill((255,255,255))
    fnt = pygame.font.SysFont("comicsans", 40)
    text = fnt.render("Time: " + format_time(time), 1, (0,0,0))
```

```
    win.blit(text, (540 - 160, 560))
    text = fnt.render("X " * strikes, 1, (255, 0, 0))
    win.blit(text, (20, 560))
    board.draw()

def format_time(secs):
    sec = secs%60
    minute = secs//60
    hour = minute//60

    mat = " " + str(minute) + ":" + str(sec)
    return mat

def main():
    win = pygame.display.set_mode((540,600))
    pygame.display.set_caption("Sudoku")
    board = Grid(9, 9, 540, 540, win)
    key = None
    run = True
    start = time.time()
    strikes = 0
    while run:

        play_time = round(time.time() - start)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                run = False
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_1:
                    key = 1
                if event.key == pygame.K_2:
                    key = 2
                if event.key == pygame.K_3:
                    key = 3
                if event.key == pygame.K_4:
                    key = 4
                if event.key == pygame.K_5:
                    key = 5
                if event.key == pygame.K_6:
                    key = 6
                if event.key == pygame.K_7:
                    key = 7
                if event.key == pygame.K_8:
                    key = 8
                if event.key == pygame.K_9:
                    key = 9
                if event.key == pygame.K_KP1:
                    key = 1
                if event.key == pygame.K_KP2:
                    key = 2
                if event.key == pygame.K_KP3:
                    key = 3
                if event.key == pygame.K_KP4:
                    key = 4
                if event.key == pygame.K_KP5:
                    key = 5
                if event.key == pygame.K_KP6:
```

```
                    key = 6
                if event.key == pygame.K_KP7:
                    key = 7
                if event.key == pygame.K_KP8:
                    key = 8
                if event.key == pygame.K_KP9:
                    key = 9
                if event.key == pygame.K_DELETE:
                    board.clear()
                    key = None

                if event.key == pygame.K_SPACE:
                    board.solve_gui()

                if event.key == pygame.K_RETURN:
                    i, j = board.selected
                    if board.cubes[i][j].temp != 0:
                        if board.place(board.cubes[i][j].temp):
                            print("Success")
                        else:
                            print("Wrong")
                            strikes += 1
                        key = None

                        if board.is_finished():
                            print("Game over")

            if event.type == pygame.MOUSEBUTTONDOWN:
                pos = pygame.mouse.get_pos()
                clicked = board.click(pos)
                if clicked:
                    board.select(clicked[0], clicked[1])
                    key = None

        if board.selected and key != None:
            board.sketch(key)

        redraw_window(win, board, play_time, strikes)
        pygame.display.update()

main()
pygame.quit()
```