In [1]:

```python
import turtle
import time
import random

delay = 0.1
score = 0
high_score = 0

# Creating a window screen
wn = turtle.Screen()
wn.title("Snake Game")
wn.bgcolor("blue")

# the width and height can be put as user's choice
wn.setup(width=600, height=600)
wn.tracer(0)

# head of the snake
head = turtle.Turtle()
head.shape("square")
head.color("white")
head.penup()
head.goto(0, 0)
head.direction = "Stop"

# food in the game
food = turtle.Turtle()
colors = random.choice(['red', 'green', 'black'])
shapes = random.choice(['square', 'triangle', 'circle'])
food.speed(0)
food.shape(shapes)
food.color(colors)
food.penup()
food.goto(0, 100)

pen = turtle.Turtle()
pen.speed(0)
pen.shape("square")
pen.color("white")
pen.penup()
pen.hideturtle()
pen.goto(0, 250)
pen.write("Score : 0  High Score : 0", align="center",
          font=("candara", 24, "bold"))


# assigning key directions
def group():
    if head.direction != "down":
        head.direction = "up"


def godown():
    if head.direction != "up":
        head.direction = "down"


def goleft():
    if head.direction != "right":
        head.direction = "left"


def goright():
    if head.direction != "left":
        head.direction = "right"


def move():
    if head.direction == "up":
        y = head.ycor()
        head.sety(y + 20)
    if head.direction == "down":
        y = head.ycor()
        head.sety(y - 20)
    if head.direction == "left":
        x = head.xcor()
        head.setx(x - 20)
    if head.direction == "right":
        x = head.xcor()
        head.setx(x + 20)


wn.listen()
wn.onkeypress(group, "w")
wn.onkeypress(godown, "s")
wn.onkeypress(goleft, "a")
wn.onkeypress(goright, "d")

segments = []

# Main Gameplay
while True:
    wn.update()
    if head.xcor() > 290 or head.xcor() < -290 or head.ycor() > 290 or head.ycor() < -
        time.sleep(1)
        head.goto(0, 0)
        head.direction = "Stop"
        colors = random.choice(['red', 'blue', 'green'])
        shapes = random.choice(['square', 'circle'])
        for segment in segments:
            segment.goto(1000, 1000)
        segments.clear()
        score = 0
        delay = 0.1
        pen.clear()
        pen.write("Score : {} High Score : {} ".format(
            score, high_score), align="center", font=("candara", 24, "bold"))
    if head.distance(food) < 20:
        x = random.randint(-270, 270)
        y = random.randint(-270, 270)
        food.goto(x, y)

        # Adding segment
        new_segment = turtle.Turtle()
        new_segment.speed(0)
        new_segment.shape("square")
        new_segment.color("orange")  # tail colour
        new_segment.penup()
        segments.append(new_segment)
        delay -= 0.001
```

```python
            score += 10
            if score > high_score:
                high_score = score
            pen.clear()
            pen.write("Score : {} High Score : {} ".format(
                score, high_score), align="center", font=("candara", 24, "bold"))
        # Checking for head collisions with body segments
        for index in range(len(segments) - 1, 0, -1):
            x = segments[index - 1].xcor()
            y = segments[index - 1].ycor()
            segments[index].goto(x, y)
        if len(segments) > 0:
            x = head.xcor()
            y = head.ycor()
            segments[0].goto(x, y)
        move()
        for segment in segments:
            if segment.distance(head) < 20:
                time.sleep(1)
                head.goto(0, 0)
                head.direction = "stop"
                colors = random.choice(['red', 'blue', 'green'])
                shapes = random.choice(['square', 'circle'])
                for segment in segments:
                    segment.goto(1000, 1000)
                segment.clear()

                score = 0
                delay = 0.1
                pen.clear()
                pen.write("Score : {} High Score : {} ".format(
                    score, high_score), align="center", font=("candara", 24, "bold"))
        time.sleep(delay)

wn.mainloop()

# import required modules
import turtle
import time
import random

delay = 0.1
score = 0
high_score = 0

# Creating a window screen
wn = turtle.Screen()
wn.title("Snake Game")
wn.bgcolor("blue")
# the width and height can be put as user's choice
wn.setup(width=600, height=600)
wn.tracer(0)

# head of the snake
head = turtle.Turtle()
head.shape("square")
head.color("white")
head.penup()
head.goto(0, 0)
head.direction = "Stop"

```

```python
# food in the game
food = turtle.Turtle()
colors = random.choice(['red', 'green', 'black'])
shapes = random.choice(['square', 'triangle', 'circle'])
food.speed(0)
food.shape(shapes)
food.color(colors)
food.penup()
food.goto(0, 100)

pen = turtle.Turtle()
pen.speed(0)
pen.shape("square")
pen.color("white")
pen.penup()
pen.hideturtle()
pen.goto(0, 250)
pen.write("Score : 0  High Score : 0", align="center",
          font=("candara", 24, "bold"))


# assigning key directions
def group():
    if head.direction != "down":
        head.direction = "up"


def godown():
    if head.direction != "up":
        head.direction = "down"


def goleft():
    if head.direction != "right":
        head.direction = "left"


def goright():
    if head.direction != "left":
        head.direction = "right"


def move():
    if head.direction == "up":
        y = head.ycor()
        head.sety(y + 20)
    if head.direction == "down":
        y = head.ycor()
        head.sety(y - 20)
    if head.direction == "left":
        x = head.xcor()
        head.setx(x - 20)
    if head.direction == "right":
        x = head.xcor()
        head.setx(x + 20)


wn.listen()
wn.onkeypress(group, "w")
wn.onkeypress(godown, "s")
wn.onkeypress(goleft, "a")
wn.onkeypress(goright, "d")

segments = []

# Main Gameplay
while True:
    wn.update()
    if head.xcor() > 290 or head.xcor() < -290 or head.ycor() > 290 or head.ycor() < -
        time.sleep(1)
        head.goto(0, 0)
        head.direction = "Stop"
        colors = random.choice(['red', 'blue', 'green'])
        shapes = random.choice(['square', 'circle'])
        for segment in segments:
            segment.goto(1000, 1000)
        segments.clear()
        score = 0
        delay = 0.1
        pen.clear()
        pen.write("Score : {} High Score : {} ".format(
            score, high_score), align="center", font=("candara", 24, "bold"))
    if head.distance(food) < 20:
        x = random.randint(-270, 270)
        y = random.randint(-270, 270)
        food.goto(x, y)

        # Adding segment
        new_segment = turtle.Turtle()
        new_segment.speed(0)
        new_segment.shape("square")
        new_segment.color("orange")  # tail colour
        new_segment.penup()
        segments.append(new_segment)
        delay -= 0.001
        score += 10
        if score > high_score:
            high_score = score
        pen.clear()
        pen.write("Score : {} High Score : {} ".format(
            score, high_score), align="center", font=("candara", 24, "bold"))
    # Checking for head collisions with body segments
    for index in range(len(segments) - 1, 0, -1):
        x = segments[index - 1].xcor()
        y = segments[index - 1].ycor()
        segments[index].goto(x, y)
    if len(segments) > 0:
        x = head.xcor()
        y = head.ycor()
        segments[0].goto(x, y)
    move()
    for segment in segments:
        if segment.distance(head) < 20:
            time.sleep(1)
            head.goto(0, 0)
            head.direction = "stop"
            colors = random.choice(['red', 'blue', 'green'])
            shapes = random.choice(['square', 'circle'])
            for segment in segments:
                segment.goto(1000, 1000)
            segment.clear()


            score = 0
            delay = 0.1
            pen.clear()
            pen.write("Score : {} High Score : {} ".format(
                score, high_score), align="center", font=("candara", 24, "bold"))
    time.sleep(delay)

wn.mainloop()

```

```
---------------------------------------------------------------------------
Terminator                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_1824/2513247795.py in <module>
     91 # Main Gameplay
     92 while True:
---> 93     wn.update()
     94     if head.xcor() > 290 or head.xcor() < -290 or head.ycor() > 290
or head.ycor() < -290:
     95         time.sleep(1)

~\anaconda3\lib\turtle.py in update(self)
   1302         self._tracing = True
   1303         for t in self.turtles():
-> 1304             t._update_data()
   1305             t._drawturtle()
   1306         self._tracing = tracing

~\anaconda3\lib\turtle.py in _update_data(self)
   2645
   2646     def _update_data(self):
-> 2647         self.screen._incrementudc()
   2648         if self.screen._updatecounter != 0:
   2649             return

~\anaconda3\lib\turtle.py in _incrementudc(self)
   1291         if not TurtleScreen._RUNNING:
   1292             TurtleScreen._RUNNING = True
-> 1293             raise Terminator
   1294         if self._tracing > 0:
   1295             self._updatecounter += 1

Terminator:
```

In [ ]:

```python

```

In [ ]:

```python

```

In [ ]:

```python

```