

In [1]:

```
1 import turtle
2 import time
3 import random
4
5 delay = 0.1
6 score = 0
7 high_score = 0
8
9 # Creating a window screen
10 wn = turtle.Screen()
11 wn.title("Snake Game")
12 wn.bgcolor("blue")
13
14 # the width and height can be put as user's choice
15 wn.setup(width=600, height=600)
16 wn.tracer(0)
17
18 # head of the snake
19 head = turtle.Turtle()
20 head.shape("square")
21 head.color("white")
22 head.penup()
23 head.goto(0, 0)
24 head.direction = "Stop"
25
26 # food in the game
27 food = turtle.Turtle()
28 colors = random.choice(['red', 'green', 'black'])
29 shapes = random.choice(['square', 'triangle', 'circle'])
30 food.speed(0)
31 food.shape(shapes)
32 food.color(colors)
33 food.penup()
34 food.goto(0, 100)
35
36 pen = turtle.Turtle()
37 pen.speed(0)
38 pen.shape("square")
39 pen.color("white")
40 pen.penup()
41 pen.hideturtle()
42 pen.goto(0, 250)
43 pen.write("Score : 0 High Score : 0", align="center",
44          font=("candara", 24, "bold"))
45
46
47 # assigning key directions
48 def group():
49     if head.direction != "down":
50         head.direction = "up"
51
52
53 def godown():
54     if head.direction != "up":
55         head.direction = "down"
56
57
58 def goleft():
59     if head.direction != "right":
```

```

60         head.direction = "left"
61
62
63 def goright():
64     if head.direction != "left":
65         head.direction = "right"
66
67
68 def move():
69     if head.direction == "up":
70         y = head.ycor()
71         head.sety(y + 20)
72     if head.direction == "down":
73         y = head.ycor()
74         head.sety(y - 20)
75     if head.direction == "left":
76         x = head.xcor()
77         head.setx(x - 20)
78     if head.direction == "right":
79         x = head.xcor()
80         head.setx(x + 20)
81
82
83 wn.listen()
84 wn.onkeypress(group, "w")
85 wn.onkeypress(godown, "s")
86 wn.onkeypress(goleft, "a")
87 wn.onkeypress(goright, "d")
88
89 segments = []
90
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() < -
95         time.sleep(1)
96         head.goto(0, 0)
97         head.direction = "Stop"
98         colors = random.choice(['red', 'blue', 'green'])
99         shapes = random.choice(['square', 'circle'])
100     for segment in segments:
101         segment.goto(1000, 1000)
102     segments.clear()
103     score = 0
104     delay = 0.1
105     pen.clear()
106     pen.write("Score : {} High Score : {}".format(
107         score, high_score), align="center", font=("candara", 24, "bold"))
108     if head.distance(food) < 20:
109         x = random.randint(-270, 270)
110         y = random.randint(-270, 270)
111         food.goto(x, y)
112
113     # Adding segment
114     new_segment = turtle.Turtle()
115     new_segment.speed(0)
116     new_segment.shape("square")
117     new_segment.color("orange") # tail colour
118     new_segment.penup()
119     segments.append(new_segment)
120     delay -= 0.001

```

```

121         score += 10
122         if score > high_score:
123             high_score = score
124         pen.clear()
125         pen.write("Score : {} High Score : {}".format(
126             score, high_score), align="center", font=("candara", 24, "bold"))
127     # Checking for head collisions with body segments
128     for index in range(len(segments) - 1, 0, -1):
129         x = segments[index - 1].xcor()
130         y = segments[index - 1].ycor()
131         segments[index].goto(x, y)
132     if len(segments) > 0:
133         x = head.xcor()
134         y = head.ycor()
135         segments[0].goto(x, y)
136     move()
137     for segment in segments:
138         if segment.distance(head) < 20:
139             time.sleep(1)
140             head.goto(0, 0)
141             head.direction = "stop"
142             colors = random.choice(['red', 'blue', 'green'])
143             shapes = random.choice(['square', 'circle'])
144             for segment in segments:
145                 segment.goto(1000, 1000)
146             segment.clear()
147
148             score = 0
149             delay = 0.1
150             pen.clear()
151             pen.write("Score : {} High Score : {}".format(
152                 score, high_score), align="center", font=("candara", 24, "bold"))
153     time.sleep(delay)
154
155 wn.mainloop()
156
157 # import required modules
158 import turtle
159 import time
160 import random
161
162 delay = 0.1
163 score = 0
164 high_score = 0
165
166 # Creating a window screen
167 wn = turtle.Screen()
168 wn.title("Snake Game")
169 wn.bgcolor("blue")
170 # the width and height can be put as user's choice
171 wn.setup(width=600, height=600)
172 wn.tracer(0)
173
174 # head of the snake
175 head = turtle.Turtle()
176 head.shape("square")
177 head.color("white")
178 head.penup()
179 head.goto(0, 0)
180 head.direction = "Stop"
181

```

```

182 # food in the game
183 food = turtle.Turtle()
184 colors = random.choice(['red', 'green', 'black'])
185 shapes = random.choice(['square', 'triangle', 'circle'])
186 food.speed(0)
187 food.shape(shapes)
188 food.color(colors)
189 food.penup()
190 food.goto(0, 100)
191
192 pen = turtle.Turtle()
193 pen.speed(0)
194 pen.shape("square")
195 pen.color("white")
196 pen.penup()
197 pen.hideturtle()
198 pen.goto(0, 250)
199 pen.write("Score : 0 High Score : 0", align="center",
200         font=("candara", 24, "bold"))
201
202
203 # assigning key directions
204 def group():
205     if head.direction != "down":
206         head.direction = "up"
207
208
209 def godown():
210     if head.direction != "up":
211         head.direction = "down"
212
213
214 def goleft():
215     if head.direction != "right":
216         head.direction = "left"
217
218
219 def goright():
220     if head.direction != "left":
221         head.direction = "right"
222
223
224 def move():
225     if head.direction == "up":
226         y = head.ycor()
227         head.sety(y + 20)
228     if head.direction == "down":
229         y = head.ycor()
230         head.sety(y - 20)
231     if head.direction == "left":
232         x = head.xcor()
233         head.setx(x - 20)
234     if head.direction == "right":
235         x = head.xcor()
236         head.setx(x + 20)
237
238
239 wn.listen()
240 wn.onkeypress(group, "w")
241 wn.onkeypress(godown, "s")
242 wn.onkeypress(goleft, "a")

```

```

243 wn.onkeypress(goright, "d")
244
245 segments = []
246
247 # Main Gameplay
248 while True:
249     wn.update()
250     if head.xcor() > 290 or head.xcor() < -290 or head.ycor() > 290 or head.ycor() < -
251         time.sleep(1)
252         head.goto(0, 0)
253         head.direction = "Stop"
254         colors = random.choice(['red', 'blue', 'green'])
255         shapes = random.choice(['square', 'circle'])
256         for segment in segments:
257             segment.goto(1000, 1000)
258         segments.clear()
259         score = 0
260         delay = 0.1
261         pen.clear()
262         pen.write("Score : {} High Score : {}".format(
263             score, high_score), align="center", font=("candara", 24, "bold"))
264     if head.distance(food) < 20:
265         x = random.randint(-270, 270)
266         y = random.randint(-270, 270)
267         food.goto(x, y)
268
269         # Adding segment
270         new_segment = turtle.Turtle()
271         new_segment.speed(0)
272         new_segment.shape("square")
273         new_segment.color("orange") # tail colour
274         new_segment.penup()
275         segments.append(new_segment)
276         delay -= 0.001
277         score += 10
278         if score > high_score:
279             high_score = score
280         pen.clear()
281         pen.write("Score : {} High Score : {}".format(
282             score, high_score), align="center", font=("candara", 24, "bold"))
283     # Checking for head collisions with body segments
284     for index in range(len(segments) - 1, 0, -1):
285         x = segments[index - 1].xcor()
286         y = segments[index - 1].ycor()
287         segments[index].goto(x, y)
288     if len(segments) > 0:
289         x = head.xcor()
290         y = head.ycor()
291         segments[0].goto(x, y)
292     move()
293     for segment in segments:
294         if segment.distance(head) < 20:
295             time.sleep(1)
296             head.goto(0, 0)
297             head.direction = "stop"
298             colors = random.choice(['red', 'blue', 'green'])
299             shapes = random.choice(['square', 'circle'])
300             for segment in segments:
301                 segment.goto(1000, 1000)
302             segment.clear()
303

```

```

304
305         score = 0
306         delay = 0.1
307         pen.clear()
308         pen.write("Score : {} High Score : {}".format(
309             score, high_score), align="center", font=("candara", 24, "bold"))
310     time.sleep(delay)
311
312 wn.mainloop()
313

```

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 []:

1

In []:

1

In []:

1

