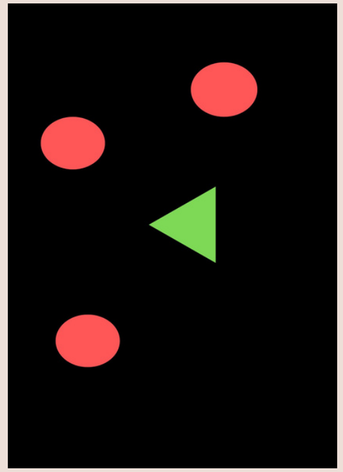
PYTHON MINI PROJECT (TEAM WORK)

TARGET TRECK

GAME DEVELOPMENT USING PYTHON

NAME OF THE PROJECT

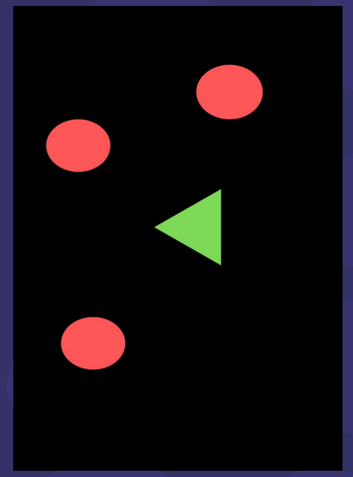
TARGET TRECK



ABOUT PYTHON

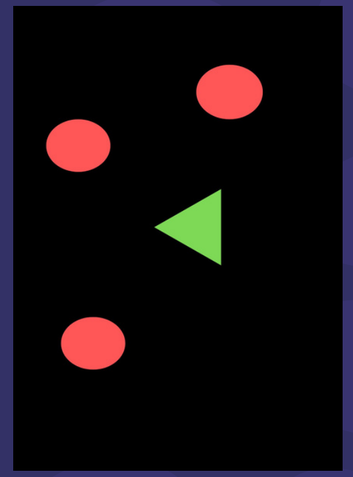
Python is an interpreted, object-oriented, high-level programming language. It incorporates modules, exceptions, dynamic typing, very high level dynamic data types and classes. Python is generally used to build websites and software, automate tasks and conduct data analysis. Python is a great programming language for game development. We now present the game we have developed using turtle module in Python.

ABOUT TURTLE MODULE



turtle is a pre-installed Python library that enables users to create pictures and shapes by providing them with a virtual canvas. The onscreen pen that you use for drawing is called the turtle and this is what gives the library its name. In short, the Python turtle library helps new programmers get a feel for what programming with Python is like in a fun and interactive way. It is mainly used for game development and creating graphical animations.

PROJECT DESCRIPTION



Our project is a simple game called "Target Treck" developed using the turtle module in Python. The game involves a player (controlled by the user) moving around a bounded area trying to collect targets (represented by circles) while avoiding hitting the boundaries of the area. The game ends if the player hits the boundary, and the score is displayed. The game has sound effects.

OBJECTIVE OF THE PROJECT

To create a simple game using the Python turtle module to demonstrate the use of programming concepts such as loops, conditionals, and functions.

MAJOR STEPS INVOLVED IN THE PROJECT

- Importing essential modules
- Setting up the screen
- Designing splash screen
- Defining game loop
- Designing game over screen

Tip: Use links to go to a different page inside your presentation.
How: Highlight text, click on the link symbol on the toolbar, and select the page in your presentation you want to connect.

PROJECT SOURCE CODE

```python
# importing essential modules
import turtle
import random
import winsound

# setting up screen
wn=turtle.Screen()
wn.setup(700,700)
wn.title("Target Treck")
wn.tracer() # avoids redrawing the screen everytime

# initial game state
game_state="start"

# defining all necessary functions in the reverse order for calling purpose
def exit():
    turtle.bye() # closes the turtle window and terminates the program

def over(n):
    turtle.Screen().clear()
    wn.bgpic("over.gif")
    screenstring="SCORE:%d"%n
    turtle.color("white")
    turtle.write(screenstring,False,align="center",font=["Arial",30,"bold"])
    turtle.hideturtle()
    wn.listen()
```

```
    wn.onkeypress(game,"C") # before game fn is called here, it would have been run once atleast, so the compiler
    wn.onkeypress(exit,"E")

def game():

    turtle.Screen().clear()
    wn.bgpic("game.gif")

    # drawing border
    mypen=turtle.Turtle() # originally at origin
    mypen.speed(0) # essential, animation speed and not movement speed, the speed with which the movements appear
    mypen.pensize(3) # shape not required
    mypen.color("white")
    mypen.hideturtle()
    mypen.penup()
    mypen.setposition(-300,-300)
    mypen.pendown()
    for side in range(4):
        mypen.forward(600)
        mypen.left(90)

    # creating player
    player=turtle.Turtle()
    player.speed(0)
    player.shape("triangle")
    player.color("lightgreen")
```

```
    player.penup()

    # creating goals
    maxgoals=5
    goals=[]
    for count in range(maxgoals):
        goals.append(turtle.Turtle())
        goals[count].speed(0)
        goals[count].shape("circle")
        goals[count].color("red")
        goals[count].penup()
        goals[count].setposition(random.randint(-270,270),random.randint(-270,270))

    # setting keyboard bindings for player
    def right():
        player.setheading(0)
    def up():
        player.setheading(90)
    def left():
        player.setheading(180)
    def down():
        player.setheading(270)
    turtle.listen()
    turtle.onkey(right,"Right")
    turtle.onkey(up,"Up")
    turtle.onkey(left,"Left")
```

```
79      turtle.onkey(down,"Down")
80
81      # game loop
82      score=0
83      while True:
84
85          # driving player
86          player.forward(10)
87
88          # boundary checking for player
89          if player.xcor()>300 or player.xcor()<-300 or player.ycor()>300 or player.ycor()<-300:
90              winsound.PlaySound("game_over.wav",winsound.SND_ASYNC)
91              break
92
93          # for each goal
94          for count in range(maxgoals):
95
96              # driving each goal
97              goals[count].forward(5)
98
99              # collision checking
100             if player.distance(goals[count]) < 20:
101                 score+=1
102                 mypen.undo() # to avoid overwriting scores
103                 mypen.hideturtle()
104                 mypen.penup()
```

```
105                 mypen.setposition(-300,300)
106                 screenstring="SCORE:%d"%score
107                 mypen.write(screenstring,False,align="left",font=["Arial",30,"bold"])
108                 winsound.PlaySound("collision.wav",winsound.SND_ASYNC)
109                 goals[count].setposition(random.randint(-270, 270), random.randint(-270, 270))
110
111             # boundary checking for each goal
112             if goals[count].xcor() > 270:
113                 winsound.PlaySound("boundary.wav",winsound.SND_ASYNC)
114                 goals[count].setx(270)
115                 goals[count].setheading(random.randint(180, 360))
116             elif goals[count].xcor() < -270:
117                 winsound.PlaySound("boundary.wav",winsound.SND_ASYNC)
118                 goals[count].setx(-270)
119                 goals[count].setheading(random.randint(0, 180))
120             elif goals[count].ycor() > 270:
121                 winsound.PlaySound("boundary.wav",winsound.SND_ASYNC)
122                 goals[count].sety(270)
123                 goals[count].setheading(random.randint(270, 450))
124             elif goals[count].ycor() < -270:
125                 winsound.PlaySound("boundary.wav",winsound.SND_ASYNC)
126                 goals[count].sety(-270)
127                 goals[count].setheading(random.randint(90, 270))
128
129     # over fn is called when the loop is terminated
130     over(score)
```

```
131
132 def start():
133     wn.bgpic("start.gif")
134     wn.listen()
135     wn.onkeypress(game,"S")
136
137 # calling start fn to start the game
138 start()
139
140 # essential, to delay and hold the screen until the user wants to close
141 delay=input("Press enter to finish")
```

NOVELTY OF THE PROJECT

- It uses simple game mechanics to create an engaging and entertaining game.
- It has a clean and minimalist design, which makes it easy to play and understand.
- The use of sound effects adds to the immersive experience of the game.

SOCIETAL USAGE OF THE PROJECT

Target Treck can be used as a fun and interactive way to teach programming concepts to beginners. Additionally, the game can be used to improve hand-eye coordination, reaction time, and problem-solving skills.

OUTCOMES OF THE PROJECT

- Knowledge of Python syntax and programming concepts
- Familiarity with Python's standard library and popular third-party modules like turtle
- Increased familiarity with development tools and environments
- Better collaboration and communication skills required for team work

TEAM MEMBERS

- ALAMELU.KR 2022506111
- ARCHANA.K 2022506128
- MEENAKSHI.C 2022506097
- NANDHITHA.K 2022506126
- NETHRA.VS 2022506115

THANK YOU!!!