```
#Undertale Battle Simulator
#AP Computer Science Principles Project

#import necessary packets
import turtle as trtl
import random as rand
import time

#make screen and set screen size
wn = trtl.Screen()
wn.setup(width=800, height=800)

#background setup
wn.bgcolor("black")

#setup sprites
sprites = ["floweyNormal.gif", "floweyEvil.gif", "undertaleDialogue.gif", "mcHeart.gif", "pellet.gif", "floweyLaugh.
gif", "sans1.gif", "sansUnderTable.gif"]
#all sprite images are taken from outside source pixelartmaker.com on January 18, 2021:
#citations:
#flowey normal - http://pixelartmaker.com/art/0179f23cf573922
#flowey evil picture - http://pixelartmaker.com/art/0535ce6cb570c5f
#heart - http://pixelartmaker.com/art/824d275f52eede8
#sans no color - http://pixelartmaker.com/art/25219d192603337
#sans with color - http://pixelartmaker.com/art/2ce07bdb4cd8e74
#pellet - http://pixelartmaker.com/art/90dc05c60e9dccf
#dialogue box - http://pixelartmaker.com/art/4f06e48d7924b0d

#add the sprites in as shapes for turtle costumes
for costume in sprites:
    wn.addshape(costume)

#boolean for whether mc is alive
dead = False

#score setup
counter_interval = 100
score = 0

#set up Flowey
flowey = trtl.Turtle()
flowey.penup()
flowey.speed(0)
flowey.shape("floweyNormal.gif")
flowey.goto(0, 200)

#set up dialogue box
talk = trtl.Turtle()
talk.penup()
talk.speed(0)
talk.shape("undertaleDialogue.gif")
talk.goto(0,-300)

#set up dialogue writer
```

```
words = trtl.Turtle()
words.hideturtle()
words.penup()
words.speed(0)
words.goto(-200,-300)
words.color("white")

#set up main character (mc)
mc = trtl.Turtle()
mc.shape("mcHeart.gif")
mc.speed(0)
mc.penup()

#Boundary Box
box = trtl.Turtle()
box.penup()
box.speed(0)
box.goto(-150, 100)
box.pendown()
box.color("white")
box.hideturtle()
#use loop to use turtle to draw a box
i = 0
while (i < 4):
    box.forward(300)
    box.right(90)
    i += 1

#scoreboard setup
scoreboard = trtl.Turtle()
scoreboard.hideturtle()
scoreboard.penup()
scoreboard.speed(0)
scoreboard.goto(200,300)
scoreboard.color("white")

#scoreboard function
def keepScore():
    global dead
    if (dead == False):
        global score
        scoreboard.clear()
        score += 1
        scoreboard.write("Score:" + str(score), font = ("Papyrus", 20, "normal"))
        scoreboard.getscreen().ontimer(keepScore, counter_interval)


#make a Sans chain to go across the screen
def sansChain(sansCostume):
    #reduce movement lag
    wn.tracer(0,0)
    #x coordinates for each sans
    sansX = [-400, -300, -200, -100, 0, 100, 200, 300, 400]
    #empty array to append the multiple sans sprites into
```

```
    manySans = []
    #get the sans into position
    i = 0
    while (i < 9):
        sans = trtl.Turtle()
        sans.hideturtle()
        sans.penup()
        if (sansCostume == "sans1.gif"):
            sans.shape(sansCostume)
        else:
            sans.shape("sansUnderTable.gif")
        sans.speed(0)
        #move sans to correct location using the array above
        sans.goto(sansX[i], 300)
        sans.showturtle()
        #after set up, add these sans into an array to use for movement
        manySans.append(sans)
        i = i + 1
    #sans movement from top to bottom of screen
    sansMove = True
    while (sansMove == True):
        for sansyboi in manySans:
            if (sansyboi.ycor() > -500):
                sansyboi.goto(sansyboi.xcor(), sansyboi.ycor() - 20)
                #update frame here
                wn.update()
            else:
                sansyboi.hideturtle()
                sansMove = False

#first Sans chain
sansChain("sans1.gif")
#second Sans chain
sansChain("If a tomato is a fruit, is ketchup a smoothie?")

#Introduction and Instructions
words.write("Hi! I'm Flowey the Flower!", font = ("papyrus", 20, "normal"))
time.sleep(2)
words.clear()
words.write("That red heart is you!", font = ("papyrus", 20, "normal"))
time.sleep(2)
words.clear()
words.write("Move using the Arrow Keys", font = ("papyrus", 20, "normal"))
time.sleep(2)
words.clear()
words.write("Don't get hit by the pellets!", font = ("papyrus", 20, "normal"))
#as game starts, change the costume of flowey to evil
flowey.shape("floweyEvil.gif")

#obstacles coordinates
xCor = [-150, -114, -78, -42, 0, 42, 78, 114, 150]
yCor = [100, 40, -20, -80, -140, -80, -20, 40, 100]
pellets = []
#obstacle setup
```

```
i = 0
#loop to give the obstacles shape and move them to a preset location
while (i < 9):
    obs = trtl.Turtle()
    obs.hideturtle()
    obs.shape("pellet.gif")
    obs.penup()
    obs.speed(0)
    x = xCor[i]
    y = yCor[i]
    obs.goto(x, y)
    obs.showturtle()
    #add the new obstacle into an array to store for future movement use
    pellets.append(obs)
    i += 1

#variable for obstacle speeds
speed = 1

#obstacle movement
def obsMove():
    wn.tracer(0,0)
    global dead, speed
    #obstacles only move while player is still alive
    while (dead == False):
        #for loop to move each obstacle that is stored in pellets
        for obs in pellets:
            obs.speed(0)
            #reset osbtacle if it is lower than the box to a random location
            if (obs.ycor() < -200):
                    obs.hideturtle()
                    #variable to hold random generated index
                    randInt = rand.randint(0,len(xCor)-1)
                    #use randInt to find the X coordinate that the resetting obstacle will go to
                    x = xCor[randInt]
                    #first go out of screen then reset position
                    #this makes sure that no accidental collisions happen during reset
                    obs.goto(-400, -400)
                    obs.goto(-400, 400)
                    obs.goto(x, 140)
                    obs.showturtle()
            #obstacle moves down at the current speed
            obs.goto(obs.xcor(), obs.ycor() - speed)
            #if collision between obstacle and mc happens, then the mc is dead
            if (obs.distance(mc.xcor(), mc.ycor()) < 15):
                obs.hideturtle()
                flowey.shape("floweyLaugh.gif")
                dead = True
            #if no collision happened and the mc is still alive, increase the speed
            else:
                speed += 0.001
                wn.update()
    #once dead, clear scoreboard and tell user their final score
    if (dead == True):
```

```
        scoreboard.clear()
        words.clear()
        words.write("YOU DIED!", font = ("papyrus", 20, "normal"))
        time.sleep(2)
        words.clear()
        words.write("Your score was: " + str(score), font = ("papyrus", 20, "normal"))

#controls for mc movement
#conditional statements to make sure mc stays inside the box at all times
def leftPress():
    if (mc.xcor() <= -140):
        mc.goto(mc.xcor(), mc.ycor())
    else:
        mc.goto(mc.xcor()-10, mc.ycor())
def rightPress():
    if (mc.xcor() >= 140):
        mc.goto(mc.xcor(), mc.ycor())
    else:
        mc.goto(mc.xcor()+10, mc.ycor())
def upPress():
    if (mc.ycor() >= 90):
        mc.goto(mc.xcor(), mc.ycor())
    else:
        mc.goto(mc.xcor(), mc.ycor()+10)
def downPress():
    if (mc.ycor() <= -190):
        mc.goto(mc.xcor(), mc.ycor())
    else:
        mc.goto(mc.xcor(), mc.ycor()-10)

#when each key is pressed, program calls a function defined above
wn.onkeypress(leftPress, "Left")
wn.onkeypress(rightPress, "Right")
wn.onkeypress(upPress, "Up")
wn.onkeypress(downPress, "Down")

#listen for key presses
wn.listen()
#start score counter
wn.ontimer(keepScore, counter_interval)
#start obstacle movement
obsMove()

wn.mainloop()
```