CodeAdvantage®
Learn Create Master

Python
Animal Expedition – Lesson 7-8
Animal object movement and borders

Student Objectives

- Create the ability for animals to move
- Define a border function to keep animals on the screen

Concepts Covered

1. Border limitations
2. For Loops
3. If/else structures
4. Multi-event handling
5. Continuous functions

Today' s Activity

Today, we will create the functionality for moving the animals as well as keeping them on the screen. We will need to finish the food and whistling commands in the final class, but we will create the structures for movement now.

Classroom Discussion

Create the animal movement functions and a border functions to keep the animals bouncing off the walls (instead of wandering off screen). We will also include the functionality for whistling and placing food though we will keep both of those global variables as False until the next class (when we will finish those functionalities).

Class Activity- Guided Work

```
def checkAnimalBorders(x,borders=gameBorders):

    if (x.xcor() >= borders['max_x'] or
        x.xcor() <= borders['min_x'] or
        x.ycor() >= borders['max_y'] or
        x.ycor() <= borders['min_y']):
        x.rt(180)
        x.fd(10)


def do_move_animals():

    if move_animals:

        for a in animal_house:

            if is_whistling:

                dist_to_player = distForm(a,player)

                move_x, move_y = 0,0

                if a.xcor() < player.xcor(): move_x = 5
                else: move_x = -5

                if a.ycor() < player.ycor(): move_y = 5
                else: move_y = -5

                #a.setheading(a.heading()+move_x+move_y)
                a.setpos(a.xcor()+move_x,a.ycor()+move_y)

            elif food_is_used:

                dist_to_food = distForm(a,food)

                move_x, move_y = 0,0

                if a.xcor() < food.xcor(): move_x = 5
                else: move_x = -5

                if a.ycor() < food.ycor(): move_y = 5
                else: move_y = -5

                #a.setheading(a.heading()+move_x+move_y)
                a.setpos(a.xcor()+move_x,a.ycor()+move_y)

            else:

                if rnum(1,100) > 50: a.rt(rnum(5,15))
                else: a.lt(rnum(5,15))
                a.fd(5)

            checkAnimalBorders(a)

        s.ontimer(do_move_animals,100)
```

Lesson Overview / Summary

1. The checkAnimalBorders() function is used to keep the animal objects on the screen by bouncing off the walls. We only need one long if statement with or statements to account for any of the 4 conditions being breached. To handle this, the turtle objects will turn around and step forward.
2. For the do_move_animals() function, first check if movement is allowed. Though this function is called to run, it continues to run using the .ontimer() method, and the move_animals variable is used to stop that continuous running when requested (when changing biomes).
3. Using a for loop, iterate over the animals in the animal_house (the ones actively showing/moving on the screen based on the current biome) and cover 3 conditions:
    - The is_whistling conditon (value is True) will cause the animals to move towards the player, altering their steps to move towards the player in each iteration that the player is still whistling.
    - The food_is_used condition (value is True) will cause the animals to alter their movement towards the food until the food is no longer visible (an animal touches it and " eats" it).
    - The 3rd (default) condition is normal motion (moving forward, turning, etc..)
4. No matter what condition is valid, ALWAYS check the border conditions to keep the animals on screen.
5. ***NOTE: Keep the is_whistling and food_is_used variables set to False until the end of the next class (where they will be changeable). Also, uncomment out the previous commented out code for running this function.

Independent Play/ Game Customization

Allow your students to enjoy their new game! Give them the last 10 minutes of class to play their game, or explore what they can do to customize their world. Celebrate creativity!

Building Relationships

Share what your students can do! Take screenshots of your student' s work, or pictures of them (absolutely no faces please!) with their computer screens, and use the handle @CodeAdvantage and the handle #CodeAdvantage to let others see and be inspired by your students' success with coding!

(Tweet or Instagram example: " My students completed @CodeAdvantege #Lets Float Around today! We designed and coded games! I' m so proud of them! My students rock!)

Consider the plans for your next lesson and start preparing for how the kids are going to have a great time with @CodeAdvantage creating #(next lesson name)!

We value your opinion! Please send any lesson suggestions, edits, comments, or questions to feedback@codeadvantage.org.