CodeAdvantage®
Learn Create Master

Python
Animal Expedition – Lesson 3-4
Player motion and Border Conditions

Student Objectives

- Define functionalities for controlled player movement
- Define the general game borders and apply to the player

Concepts Covered

1. If/elif/else structures
2. Dictionaries
3. Functions
4. Global Variables
5. Turtle object creation

Today' s Activity

Today we will work on creating out main player, mapping movement and creating a border-checking functionality that applies to the player. We will start it today but some code will be commented out until we finish more development.

Classroom Discussion

Create the main player object and start building the mechanisms needed for biome changing. The checkBorders() function will be used to change biomes (using a function we will write in the future) but today we will get the player moving and start the biome backend algorithms.

Class Activity- Guided Work

1.

```
# MAKE GAME BORDERS
gameBorders = {'min_x':-550,'max_x':550,'min_y':-225,'max_y':225}

# MAKE PLAYER
player = turtle.Turtle()
player.penup()
player.shape('turtle')
player.color('red')
player.shapesize(2,2)

# Add player shape
s.register_shape('player.gif')
player.shape('player.gif')
```

2.

```
def checkBorders(x,xs,borders=gameBorders):

    global gridX
    global gridY

    # HANDLE Y Direction -----------------------------------
    if x.ycor() >= borders['max_y']:

        if gridY < 3:
            x.hideturtle(); x.sety(borders['min_y']+20)
            gridY += 1
            set_biome(); x.showturtle()

        else:
            x.sety(borders['max_y']-20)

    elif x.ycor() <= borders['min_y']:

        if gridY > 1:
            x.hideturtle(); x.sety(borders['max_y']-20)
            gridY -= 1
            set_biome(); x.showturtle()

        else:
            x.sety(borders['min_y']+20)

    # HANDLE X DIRECTION -----------------------------------
    elif x.xcor() >= borders['max_x']:

        if gridX < 3:
            x.hideturtle(); x.setx(borders['min_x']+20)
            gridX += 1
            set_biome(); x.showturtle()

        else:
            x.setx(borders['max_x']-20)

    elif x.xcor() <= borders['min_x']:

        if gridX > 1:
            x.hideturtle(); x.setx(borders['max_x']-20)
            gridX -= 1
            set_biome(); x.showturtle()

        else:
            x.setx(borders['min_x']+20)
```

3.

```
def m_up():     player.sety(player.ycor()+10); checkBorders(player,s)
def m_left():   player.setx(player.xcor()-10); checkBorders(player,s)
def m_right():  player.setx(player.xcor()+10); checkBorders(player,s)
def m_down():   player.sety(player.ycor()-10); checkBorders(player,s)

s.onkeypress(m_up,'Up')
s.onkeypress(m_left,'Left')
s.onkeypress(m_right,'Right')
s.onkeypress(m_down,'Down')
```

Lesson Overview / Summary

1. Using [#1], create the game_borders dictionary to hold the limits of the game screen (where the borders for the player are to change screens and where the borders for everyone is in the event that the object must be bound to stay on the screen.
2. ***NOTE: Comment out the 2 lines underneath >> #Add Player shape << until the player.gif image has been created by the students.
3. Using [#2], create the checkBorders() function, which is designed to watch the player and update the global variables for the current gridX and gridY positions (used to parse the biome dictionary key, value pairs) and will handle whether the biome should be changed (calls set_biome() – which will be defined in the future) or bounces off the walls if a limit has been reached (i.e. we cannot go to 0,1 or 4,3, etc..). Handle all 4 directions separately.
4. ***NOTE: Comment out the lines that read <<set_biome() >> until AFTER the set_biome() function and all related code is completed (this will be noted throughout the lesson plans).
5. Using [#3], map the button controls to the player and apply the checkBorders() functionality to each one. For now, the borders will do nothing but change the global variables but will be essential in the game design after the next lessons are completed. Writing this now keep the design uniform and sets up functionality for later use.

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.