YEAR 11 INFORMATION TECHNOLOGY

ALICE WORKSHOP: OBJECT FUNCTIONS AND MORE VARIABLES

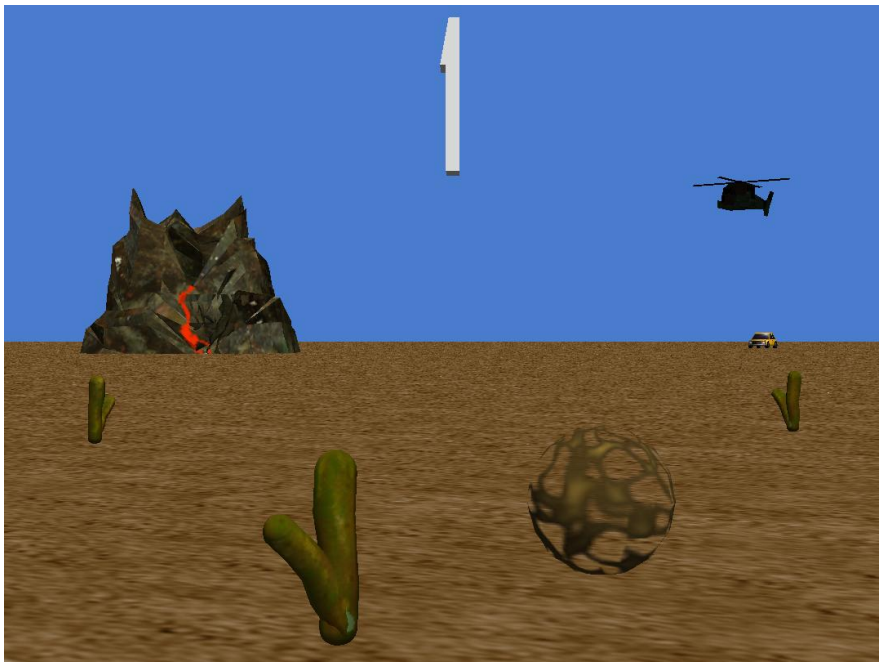
LEARNING OBJECTIVES

The learning objectives for this workshop are:

- Students can design and implement programs using object variables and global variables
- Students can use object-level functions

EXERCISE 1

Create a new world that includes a car (Vehicles gallery), armyCopter (Vehicles gallery) and 3D text as shown in the picture below. Set the text for the 3D text object to "0" and rename the object



"OdometerDisplay".

In this exercise you will use an object variable to keep track of how far the car has moved.

1. Create a new object (property) variable, odometer,with a Number type, for the car object. Initially this variable should be set to 0. To do this, select the car object, then click on the properties tab and click on the *create new variable* button.
2. Create a new method, nameddriveForward, for the car object.
    a. Create a parameter, *distance*, representing how far the car will drive forward

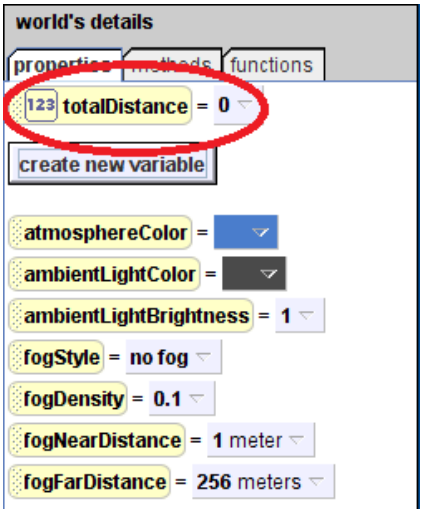
b. Add an instruction to drive the car forward *distance*metres.
c. At the same time update the *odometer* variable by adding *distance*metres.
d. In my first method, call the driveForward command several times with different values passed to the distance parameter. Select the car object and click on the properties tab so that the odometer variable is displayed. Run the program and observe the changes to the odometer variable. It should increase with each call to the driveForward command.
e. Change the time that it takes to update the odometer variable to 1 second (by default this will be set to 0 seconds). Rerun the program, you should now notice that the variable changes continuously as the car drives forward.

3. In this step you will display the distance moved by the car in the 3D text object.
   a. Create a new object method, displayOdometer, for the car object. This method should set the text property of the OdometerDisplay 3D text the current value of the *odometer* variable. Set the duration to 0.02 seconds.
   b. Call displayOdometer in my first method. This should be called within an infinite loop that runs at the same time as the calls to the driveForward method. This ensures that the odometer display is continuously updated. (Note: we will discuss a better way of doing this when events are introduced in a later lesson).

EXERCISE 2

This exercise continues Exercise 1, so make sure you have finished this before attempting this one. In this exercise you will use a global variable to keep track of the total distance travelled by the car and the helicopter.

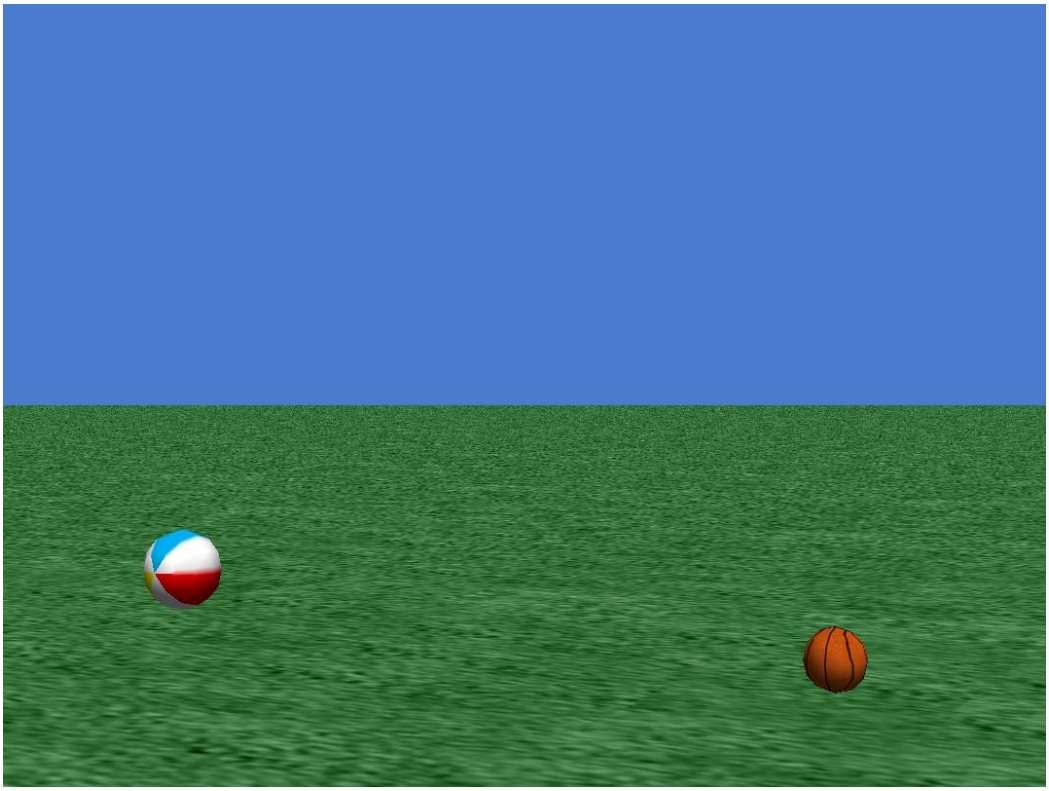
1. Create a new global variable in the world object, called *totalDistance*. Set the initial value to 0 (see screenshot).



2. Create a new method for the world object called **updateTotalDistance**. This method should include a parameter, *distance*. The method should update the *totalDistance* variable by adding *distance* to the previous value. The duration for this command should be 0 seconds.
3. Update the **driveForward** method for the car object so that it calls **updateTotalDistance**.
4. Create a new method, fly, for the armyCopter object. This method should include a distance parameter. The method should move the helicopter forward and call **updateTotalDistance**.
5. Update my first method so that the helicopter flies at the same time that the car drives forward.
6. Select the properties tab of the world object and then run the world. You should observe the *totalDistance* variable being updated as the car and helicopter move forward.

EXERCISE 3

Create a new world with a beachball and basketball (from the Sports gallery) as shown below.



In this exercise you will use object functions to move the basketball towards the beachball until they are on contact. All code can be placed in **my first method**.

1. Turn the basketball towards the beachball, and orient the beachball to the basketball.
2. Create a new local variable, *distance*, with initial value of 0.
3. Set *distance* to equal the distance between the two balls. Hint: you should use the **distance to** function, however this measure the distance between the centres, so you should subtract the radius of the each ball to compensate. The radius is equal to half of the height (or width) of the ball.
4. Move the basketball forward by *distance*metres.
5. Move the basketball backward by half *distance* metres. At the same time move the beachball forward half *distance*metres.

EXTENSION

6. Modify the code so that the basketball bounces 5 times as it moves towards the beachball, instead of just moving straight ahead. Hint: to make the basketball bounce you will need to move it up then down a small height at the same time as moving it forward. To determine how far to move forward use the *distance* variable and divide by the number of bounces.