Exercise 2.4 Keep a moving object within the screen Week 5

In the previous exercise you made the ball move over the screen, but the code was written in such a way that the ball moved from left to right, then stopped at the end of the graphical window. When simulating a less regulated movement of a ball the path is less, or even un-predictable and containing the ball inside the graphical window should be programmed in a different way.

2.4a

The algorithm for bouncing is very simple: when the ball reaches the top or bottom border of the graphical window, change the sign of the displacement in vertical direction (dY = -dY). When the ball bounces to the left or right side of the window, change the sign of the horizontal displacement (dX).

Write a program in which a ball moves in a diagonal line at constant speed and bounces when it reaches one of the borders of the graphical window. The ball should keep moving until the graphical window is closed. Try different angles of movement.

2.4b

Extend your program from exercise 2.4a. Write a program that simulates the bouncing of a ball in the presence of gravity: decrease the ball displacement in the vertical direction when the ball is moving upwards and increase the ball displacement in the vertical direction when the ball is moving downwards by subtracting a small value from dY (e.g. dY=dY-0.1).
If the ball bounces higher each step, adjust the program so that the peaks are lower each the time the ball bounces.

Exercise 2.5 Random particle diffusion Week 5

In this exercise you will simulate the random movement of a particle.

2.5a

Generate random values for dX and dY between (for instance) -5 and 5. The particle should not be allowed to move outside the graphical window. You can change the speed at which the particle moves by varying the range of your random values.

2.5b

Extend your program from exercise 2.5a. Instead of one particle, make a program with 100 particles and make them move randomly, similar to the particle in the previous programs of this exercise.
TIP1*: You'll need an array with the x-coordinates for the* particles*, and an array with the y-coordinates for the particles.*
TIP2: *Perform separate tasks in separate loops.*
TIP3: *When you update your* graphical window *for every particle individually your program will slow down, to avoid this first change the coordinates of all particles and draw them to the* graphical window *before updating it.*

Exercise 2.6 Particle collision Week 5

2.6a

Write a program where two different types of particles diffuse randomly within the graphical window. Use ten of each where each type has its own colour, size and speed.

TIP*: store the properties of each particle in an array. Because the properties are different types of variables, create an array for each property. For instance an array with a length of 20 of the type* int *to store the sizes of all 20 particles in, an array of the type* Color *to store the colour of all the 20 particles in, etc..*

2.6b

Extend your program from exercise 2.6a.
Write a program in which particles seize to move when they meet/collide with (distance less than 15 coordinate units) a particle of the other type.
Use the following equation to calculate the distance *d* between two particles *P* and *Q*:

$$d = \sqrt{(Px - Qx)^2 + (Py - Qy)^2}$$

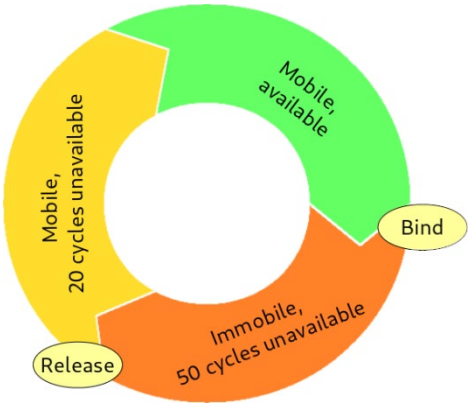
TIP1: *In case it takes too long before particles stop, increase the distance for collision.*
TIP2: *In exercise 2.7a you created multiple arrays in which to store the properties of each particle. In that exercise it might seem convenient to create two different arrays per particle type, but would that still be convenient if you have e.g. 300 different types of particles, all interacting with each other? Instead try to see the different types of particles just as particles that happen to have the 'property' of being of a different type. So create an array in which to store the type of each particle and create one single array to store the sizes of all the particles, one array to store the shapes of all the particles, one colour array, etc.*

2.6c

Extend your program from exercise 2.6b.
Write a program that only stops particles from moving when they meet a particle of the other type that is moving as well (distance less than 5 coordinate units). Increase the number of particles and change the ratio between the two types to 40:60.
TIP: *use an extra array in which to store the property of whether or not a particle is moving.*

2.6d

Extend your program from exercise 2.6c.
Now, if a particle binds (meets) a particle of the other type it should change colour, stop moving and stay bound for 50 time steps. After those 50 time steps the particles should again change colour and start moving again, but remain unable to bind again for another 20 time steps. After those 20 time steps the particles should go back to their original colour and be able to bind again.



2.6e

Extend your program from exercise 2.6c.
Let particles bind with a probability of 0.1 per time step and release with a probability of 0.01 per time step.
Notice that you will need to know which particles are bound together to release them both at the same time step.