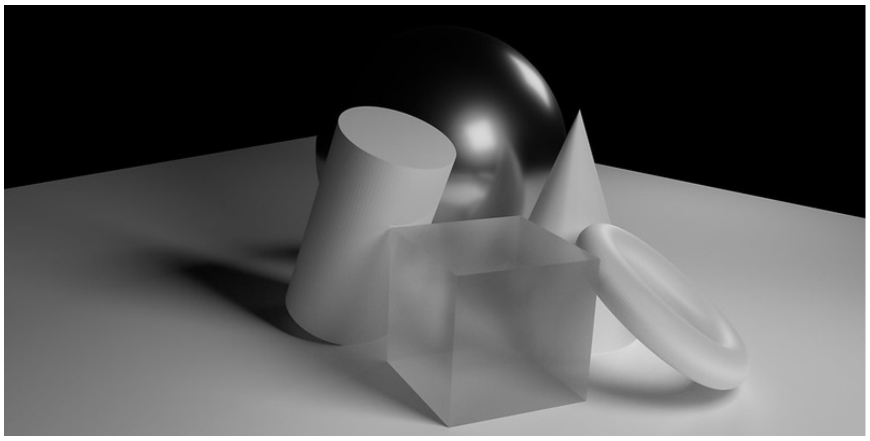
Snap! XY Geometry and 3D effects

UBMS Game and Robo Programming with Snap! and Python

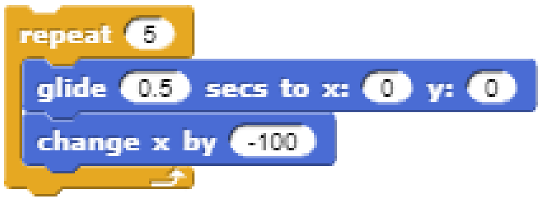
June 27, 2023



XY Geometry and Motion

- Open a new project `Geometry` in the cloud Snap! editor
- To see the grid, change the background to `XY-grid`
- Every point on the screen has two coordinates, `x` and `y`
- Every sprite in Snap! has a *center point* which you can view and change in the paint editor
- Open the `Paint editor` on the standard turtle sprite
- Turn the turtle into a **solid red square** in the center
- Go to `Scripts` and move the sprite to the exact position (0,0)

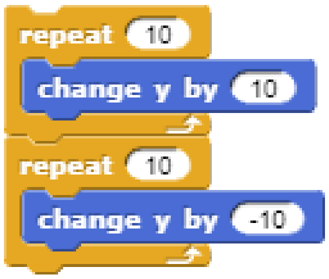
- Move it to (-200,100)
- Let it `glide` in `2 secs` to the center position (0,0)
- Add to the last command by changing the `x` position by `-200`
- Add to the last commands and let the square glide `5` times in `0.5 sec` back and forth between these the last two positions:



```
repeat 5
    glide 0.5 secs to x: 0 y: 0
    change x by -100
```

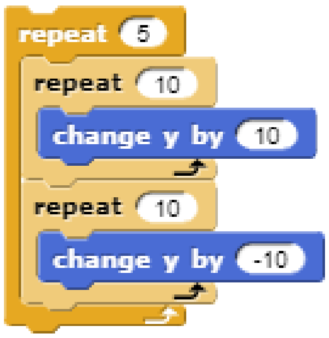
- Duplicate the last program and make the square jump up and down **smoothly** between the positions (0,0) and (0,100) ten times **without** using the `glide` command.

  Tip: smooth motion is a combo of `repeat` *and* `change`.



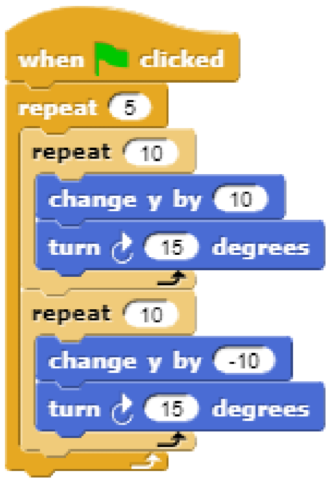
```
repeat 10
    change y by 10
repeat 10
    change y by -10
```

- Add a command to the last program to `repeat` the jump `5` times - this is called a *nested loop*.



```
repeat 5
    repeat 10
        change y by 10
    repeat 10
        change y by -10
```

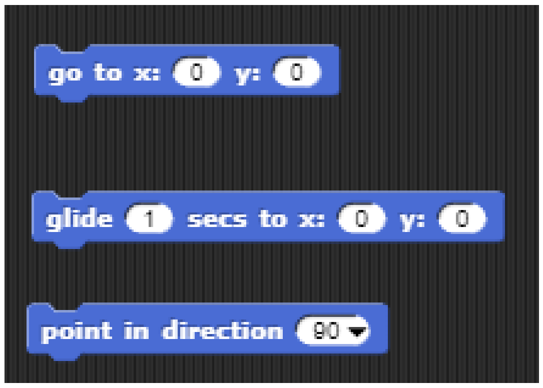
- Can you combine rotating and jumping as shown in this project? __



```
when [flag] clicked
repeat (5)
    repeat (10)
        change y by (10)
        turn ↻ (15) degrees
    repeat (10)
        change y by (-10)
        turn ↻ (15) degrees
```

Absolute motion

- Absolute motion is independent on current position and direction:
    1. motion commands yield a vector that ends at a fixed point no matter where the sprite is before the command.
    2. The movement looks different depending on where the sprite was before the command.
- Example: "Go to the Salty Dog" (That's the Lyon College café)
- Snap! commands that describe absolute motion:



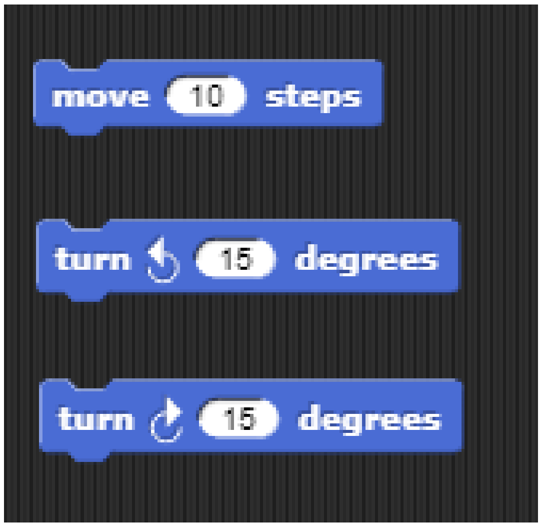
```
go to x: (0) y: (0)

glide (1) secs to x: (0) y: (0)

point in direction (90 ▾)
```

Relative motion

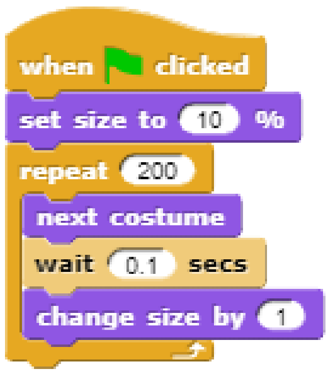
- Relative motion depends on current position and direction:
  1. motion commands yield a vector that could be anywhere in the plane.
  2. The movement looks the same no matter where the sprite was before the command.
- Example: "Turn right then left"
- Snap! commands that describe relative motion:



Practice 3-D effect using looping

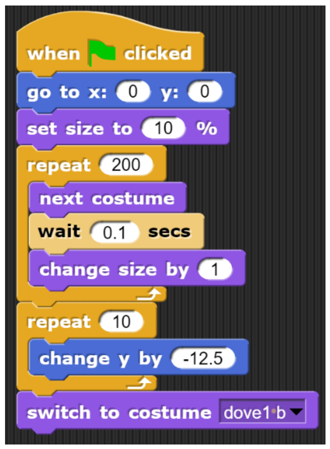
- Sprite is fully 2-dimensional
- For 3D effects you need to create illusions
- Repetition and size change can do that
- Create a new project and call it `3-dimensions`
- Load the costumes `dove1 a` and `dove1 b` on the same sprite

- Load any outdoor or indoor background as the stage
- Create and run the following script:



```
when [flag] clicked
set size to (10) %
repeat (200)
    next costume
    wait (0.1) secs
    change size by (1)
```

- Modify the script so that the dove slowly sinks to the floor and stays there when it has arrived in the foreground, with its wings down.



```
when [flag] clicked
go to x: (0) y: (0)
set size to (10) %
repeat (200)
    next costume
    wait (0.1) secs
    change size by (1)
repeat (10)
    change y by (-12.5)
switch to costume [dove1-b]
```

Program 3: Flying bat

- Create a new project called `Flying_bat`
- From the Snap! online library, load a bat costume
- Make the sprite change costume so that it looks as if its wings flap up and down.
- Add code so that when the script is started, the bat flies to the edge of the screen, bounces back, and changes direction forever.
- Optionally, combine the 3D effect and make the bat not just fly from left to right but from the background to the foreground as well.
- Upload the program to Canvas