

Parameters

Alice

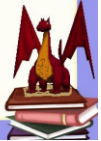


Parameters

- Built-in methods provide **flexibility** by providing parameters such as distance and direction.
 - Parameters allow you to pass in values (**arguments**).
- ✎ Example

```
georgeBeetle ▾ move up ▾ 0.5 meters ▾ duration = 0.5 seconds ▾ m
```

Parameters: distance, direction
Arguments: 0.5 meters, 0.5 seconds



Passing Arguments

- Methods are written to accept **arguments**
- Arguments are values that are passed

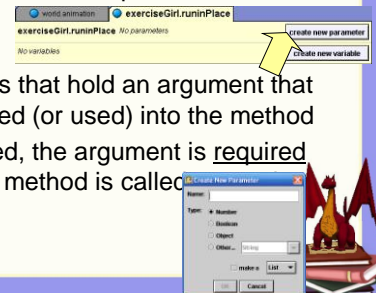
✎ Example of arguments:
– Direction, distance traveled

5-3



Passing Arguments

- Passing arguments requires the use of parameters
- Parameters are variables that hold an argument that will be passed (or used) into the method
- Once created, the argument is **required** anytime the method is called



5,4

Passing Arguments Example



- Rather than have ExerciseGirl runInPlace X times, ...
 - ✎ a parameter could be created called *repetitions*
 - ✎ would allow the viewer to enter an amount ExerciseGirl should run in place



Object Parameters

- Some parameters can be an object since some methods use objects as an argument
 - ✎ **turn to face** is a method with an object as an argument
 - ✎ **move to** is another example
- When creating an object parameter, identify the new parameter as an **object**



5-6

Parameters

- Allow methods to be “generic” i.e. reusable with different values
- Types are
 - Number
 - Boolean
 - Object
 - Other (such as sound or color)



Parameters

- The value to be used by the method is sent via an argument in the method call
- The “calling” method sends the specific value
- The method executes its code using that value



Parameter

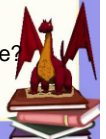
- A solution is to write a class-level method with an object parameter that allows you to pass in the specific object.

```
cleverSkater.skateAround
Parameter: whichObject
Do in order
Do together
    cleverSkater turn to face whichObject
    cleverSkater lift right leg
    cleverSkater move to whichObject
    cleverSkater turn around whichObject
```



Translation to Code

- Most of the *skateAround* storyboard design is straightforward and easy to code.
- One step, however, requires some thought:
cleverSkater move to whichObject --
what distance should the cleverSkater move?



Game Done at 10

Alice



Example

- **WacAMole** arcade game.
- Little moles pop up from holes in the top of the booth. The user tries to whack the mole before it drops out of sight.
- The game is over when 10 moles are whacked.



Designing the game

- To design the game animation, we need to answer several questions:
 - How will the game work, overall?
 - How do we keep score?
 - How do we know when the user whacks a mole?
 - How will a list help us?

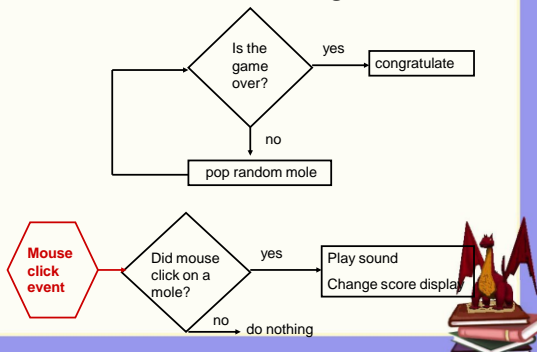


How will the game work, overall?

- A mole pops up and then goes back down. Each time the mole pops up, the user attempts to use the mouse to click the mole.
- When a click occurs, we check to see if a mole was clicked. If so, the score increases.
- The above actions repeat until the game is over.



Overall design



Storyboard for overall game

- This is the main driver for the game.
- The code will be written in *World.run*

While the game isn't over
randomly select one mole and call *popMole*
Congratulate the user



Storyboard: *popMole*

popMole

Do in order

- Move the mole up
- Wait some time
- Move the mole back down



How do we keep score?

- We will use a visual scorekeeper.
- The scorekeeper is made up of two cylinders
 - A gray cylinder above the ground
 - A yellow cylinder below the ground.
- Each cylinder is 1 meter in height.



How do we keep score?

- Each time the user successfully clicks a mole, the yellow cylinder will move up 1/10 meter.
- When the yellow cylinder is above ground (has moved up 10 times), the game is over.



How do we know when the user has clicked on a mole?

- This is where a list comes in handy:
 - **create a list of the moles** (one mole is below each hole)
 - **create a mouse click event**
 - **each time the mouse is clicked, call a score method to iterate through the list** of moles to see whether one of the moles has been clicked!



Storyboard: keep score

Event: User clicks mouse
Response: *score*

For All in order

If any mole in the mole list was the object clicked
Move the *playerscore* (yellow cylinder) up 1/10 meter

