Creating an Animation Program

Alice
Step 1: Design

- Decide on the problem to be solved
- Design a solution

- We will use a storyboard design technique, commonly used in the film industry.
Example

The **scenario** is:

Several snowpeople are outdoors, on a snow-covered landscape. A snowman is trying to meet a snowwoman who is talking with a group of her friends (other snowwomen.) He says “Ahem" and blinks his eyes, trying to get her attention.

The **problem** is:

How can we create this animation?
Create Initial World
Storyboard
Option 1: Sketches
Storyboard
Option 2: Screen shots

Initial scene

Snowman tries to catch snowwoman’s attention

Snowwoman looks around
A textual storyboard is like a "to-do" list.

The Learning to Program in Alice textbook puts a textual storyboard in a box:

Do the following actions in order:
- snowman turns to face snowwoman
- snowman "blinks eyes" and calls out to the snowwoman.
- snowwoman turns around.
Step 2: Implementation

To implement the storyboard, translate the actions in the storyboard to a program.

**Program** (a.k.a. **script**)

- a **list of instructions** to have the objects perform certain actions in the animation
Action Blocks in Alice

Sequential Action Block
- actions occur one after another

Simultaneous Action Block
-- actions occur at the same time
Demo

Ch02Snowpeople
Concepts in this first program

Program instructions may have arguments

Example: for the move instruction, the arguments we used in this example were

- direction
- distance

DoTogether and DoInOrder blocks can be nested one inside the other
Testing

An important step in creating a program is to run it – to be sure it does what you expect it to do.

We recommend that you use an **incremental development** process:

- write a few lines of code and then run it
- write a few more lines and run it
- write a few more lines and run it…

This process allows you to find any problems and fix them as you go along.
Comments

While Alice instructions are easy to understand, a particular combination of the instructions may perform an action that is not immediately obvious.

Comments are used to document the code – explain the purpose of a particular segment of the program to the human reader.
Ch02SnowpeoplewithComments

Comments in this example world illustrate

- description of the action performed by the entire method
- description of the purpose of a small segment of code
A major part of learning how to program is figuring out how to "put together the pieces" that compose a program.

Analogy:

putting together the pieces of a puzzle

The purpose of this session is to

- define the fundamental pieces of a program
- demonstrate how to put the pieces together
Four Fundamental Pieces

- Instruction
- Control Structure
- Function
- Expression
Instruction

An instruction is a statement that executes (is carried out by the computer at runtime).

In Object Oriented Programming, an instruction is defined as a method.

In Chapter 2, we used instructions to make objects perform a certain action.

Examples:
- snowman turn to face snowwoman
- spiderRobot move up 0.5 meters
A control structure is a statement that controls which instructions are executed and in what order.

In previous worlds, we used:
- Do in order
- Do together
Functions

A function asks a question (to check a condition) or computes a value.

In Alice, a function is used to determine
- the properties of objects
  - Is the snowwoman's face red?
- the relationship of one object to another
  - What is the distance between the mummy and pyramid?
- a current condition
  - What key (on the keyboard) was pressed?

Let's look at an example…
Problem Example

A Hollywood movie set. The camera angle influences our perception of the scene. Is the mummy taller than the pharaoh? How far (meters) is the mummy from the pharaoh?
Built-in Functions

Categories

- proximity
- size
- spatial relation
- point of view
- other

This example illustrates some built-in proximity functions.
Values

When a function is used to ask a question or perform a computation, an answer is returned. The answer is called a **value**.

The **type** of value depends on the kind of function.

- In our example, we want to ask the question:
  - What is the distance of the mummy to the pharaoh?
- We expect to get a number value. It could be a whole number or a fractional value, such as
  - 3 meters or 1.2 meters
Concepts illustrated in this example program:

- The built-in *distance to* function determines the distance from the center of one object to the center of another object.
- A function is not a "stand-alone" instruction; it is nested within another instruction.
Types of Values

In our example, we used a function that has a number value. Other types of values include:

- **Boolean**
  - `true`, `false`

- **String**
  - "Oh, Yeah!"

- **Object**
  - `snowman`, `helicopter`

- **Position in the world**
  - `(0, 0, 0)` – the center of an Alice world
Problem: Collision

- When the program is run, the mummy collides with the pharaoh.
- The problem is the distance between two objects is measured center-to-center.
- One way to avoid a collision is to subtract a small number (1 or 2) from the distance.
Expressions

An expression is a math or logic operation on numbers or other types of values.

Alice provides math operators for common math expressions:

- addition \(+\)
- subtraction \(-\)
- multiplication \(*\)
- division \(/\)
Concept illustrated in this example:

- Math expressions are created within an instruction.
- Available math operators are +, -, *, /
Subtracting 2 meters from the distance is an arbitrary amount.

To be more precise, we could subtract the width of the pharaoh.

The resulting expression subtracts the value of one function from the value of another function.
To be even more precise, we could subtract half of the sum of the pharaoh’s and mummy’s width. That is, 
\[
\frac{(\text{width of the pharaoh} + \text{width of mummy})}{2}
\]

Now, the expression gives a very close estimate of the distance to move.
Assignment

- Read Chapter 2 sections 1 and 2
  - Scenarios and Storyboards
  - A First Program
- Read Tips & Techniques 2
  - Orientation and Movement Instructions
- Read Chapter 3 Section 1, Functions and Expressions