Lecture 3

Object-Oriented Programming

Design & Storyboarding
Implementation & Methods

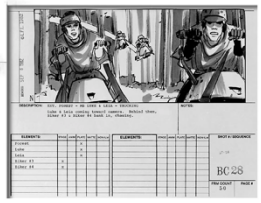
CSCI 053
Department of Computer Science
The George Washington University
Spring, 2010

Step 1: Design

Decide on the problem to be solved
define user story

Design a solution

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



Example

The **scenario** is: nouns = objects
Princess Escape verbs = actions

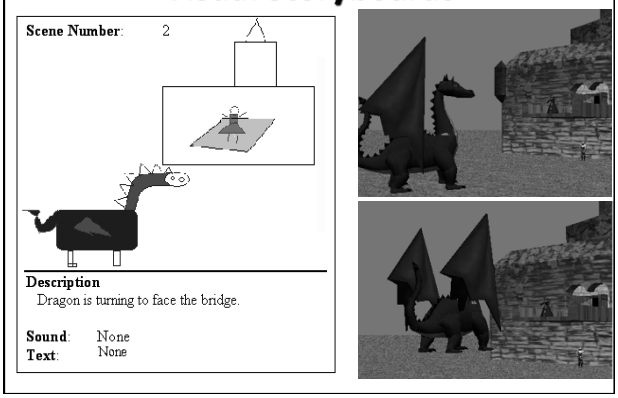
A princess has been grounded by her father (a wizard) and kept inside the castle. Being a rather rebellious princess, she has emailed the local dragon taxi service. The dragon will fly to the princess and she will climb aboard the dragon to escape from the castle – to meet some friends at the village dance club.

The **problem** is:

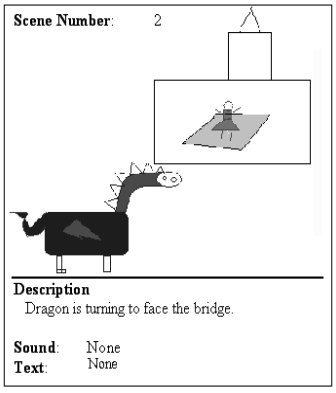
- How can we create this animation?
- What is the next step?

Create Initial World

Visual Storyboards



Storyboard Option 1: Sketches



Storyboard Option 2: Screen Shots

Initial scene

The dragon is flying towards the princess

Storyboard Option 3: Text Form

A textual storyboard is like a "to-do" list:
(*pseudo code*)

```
Do in order
    dragon takes off
    dragon flies to princess
    princess climbs on dragon's back
    dragon and princess escape
    knight shakes his arm (and sword) in protest
```

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

Writing the Program

Our planned storyboard (to-do list) is:

```
Do in order
  dragon takes off
  dragon flies to princess
  princess climbs on dragon's back
  dragon and princess escape
  knight shakes his arm (and sword) in protest
```

The idea now is to translate the design steps to program instructions.

Traditional Problem Solving in CS

1. Read and understand the problem or task specification
2. Design a solution (develop an algorithm)
3. Implement (code)
4. Test
5. Revise, as needed

Translating the Design

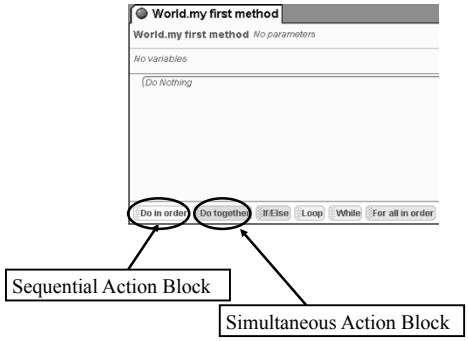
Some steps in the storyboard might be written as a single instruction

- The rabbit turns to face Alice

Other steps are composite actions that require more than one instruction

- Let's start with getting the dragon to take off

Action Blocks in Alice



Dragon Takes Off Method

We want the dragon to move up and flap its wings at the same time

Later when the dragon flies to the princess we will want it to move forward and flap its wings

We should create a flag wings method to use in both of these steps

Implementing a program in Alice

Create the program instructions using the drag and drop editor

You should use an incremental development approach

- write a method
- test it
  - and so on….

Stepwise refinement - 1

Do in order
- dragon takes off
- dragon flies to princess
- princess climbs on dragon's back
- dragon and princess escape
- knight shakes his arm (and sword) in protest

How can a dragon "take off"?

Do together
- dragon moves up
- dragon flaps wings

Stepwise refinement - 2

Do together
- dragon moves up
- dragon flaps wings

How can a dragon flap its wings?

Do together
- dragon flap left wing
- dragon flap right wing

Stepwise refinement - 3

Do together
- dragon flap left wing
- dragon flap right wing

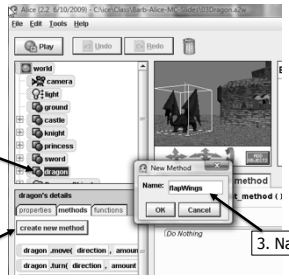
How can a dragon flap its left wing?

Do in order
- *Do together*
  - dragon left wing rolls right
  - dragon left wing flap rolls right
- *Do together*
  - dragon left wing rolls left
  - dragon left wing flap rolls left

Is this too detailed?

Probably, but you get the picture…

Create the flapWings Method



flapWings

Will flap the left wing and the right wing at the same time

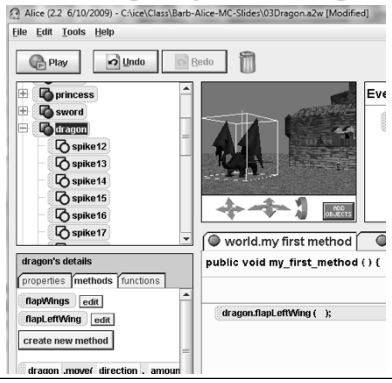
So create a flapLeftWing and a flapRightWing method as well

Write the flapLeftWing method

```
Do in order
  Do together
        dragon left wing rolls right
        dragon left wing flap rolls right
  Do together
        dragon left wing rolls left
        dragon left wing flap rolls left
```

The complete method

Testing flapLeftWing



Your turn!

Create the `flapRightWing` method

Do in order

Do together

dragon right wing rolls left 0.05

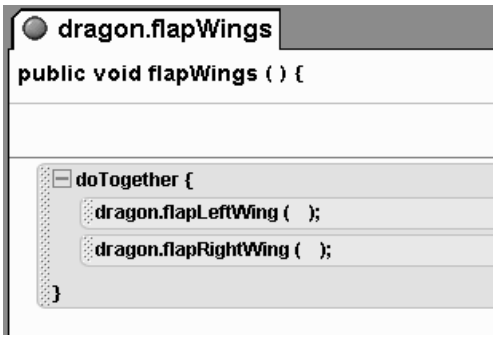
dragon right wing flap rolls left 0.03

Do together

dragon right wing rolls right 0.05

dragon right wing flap rolls right 0.03

Create a *flapWings* method



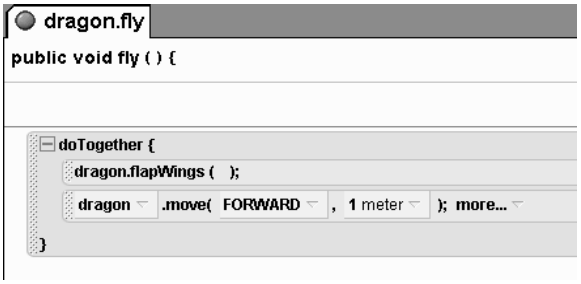
Your turn!

Create the *flapWings* method

Create a *takeOff* method, where the dragon moves up 2 meters and flaps its wings twice

- What changes will you need to make to the *duration*= parameter to get the animation working?

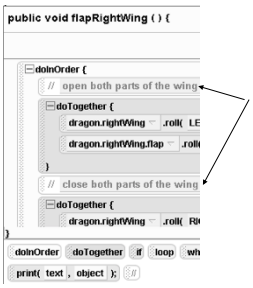
Create a fly method



Adding Comments

Comments are short descriptions of what is happening
- Explain confusing bits

Drag up the // tile
- Click on the down arrow
- Select other
- Type the comment



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.

Methods: Why use them?

Object methods:

To provide an object with additional behaviors

World methods:

To organize your story into more manageable pieces

World methods for Scenes and Shots

User stories can be divided into scenes and shots

A convenient technique for completing a project

Scene: segment of a story

→ Scenes can be divided into shots

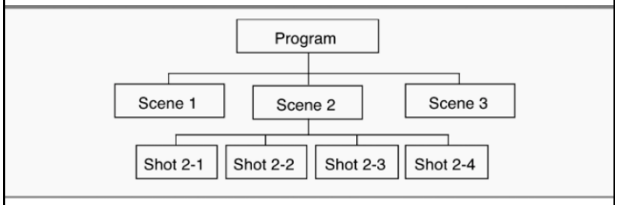
Shot: part of a scene from a given camera position

→ Shots can be further divided into pieces

Reasons for using scenes, shots, and pieces
- To create a program that reflects the user story
- To create a program that has a modular design

Divide and conquer approach to building user stories
- Break a big problem into smaller problems
- Solve each of the smaller problems
- Combine the smaller problems into a solution

Program tree



Last step: Testing your solution

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

You should use the **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.

Challenge 1: My homework grade

User story:

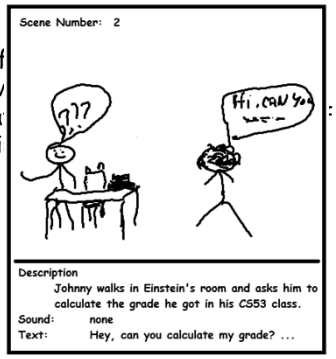
I got my grade for my late homework, and I am confused. I will to go my genius neighbor and ask him what would have my grade been if I turned in on time.

Challenge 1: My homework grade

User story:

I got my grade f
am confused. I v
and ask him wha
I turned in on ti

Storyboard:



Challenge 1: My homework grade

User story:

I got my grade for my late homework, and I am confused. I will to go my genius neighbor and ask him what would have my grade been if I turned in on time.

Storyboard:

Creating an Alice world and the objects we will need in the world

- And position them

Create a method or methods to do at least one scene in your story

Challenge 2: Over to you!

Think about a story you would like to tell using Alice

Create a storyboard for your Alice world

Create your Alice world and the objects you will need in your world

- And position them

Create a method or methods to do at least one scene in your story.

Summary

Storyboarding helps you design a story

Break the action into scenes

Break scenes into an existing method or create a new method
- That uses existing methods

The problem solving process is:
- Read and understand the problem or task specification
- Design a solution (develop an algorithm)
- Implement (code)
- Test
- Revise, as needed