CompSci 4
Chap 5 Sec 1
Oct 11, 2007

Prof. Susan Rodger

Announcements

- Read Chapter 5 Sec 2 for next Tuesday
- New groups today
- Assignment 5 out
  - Part 1 and Part 2 Due Oct. 24
- Test 1 back today

- Today
  - Interactive programming

Control of Flow

- Control of flow – how the sequence of actions in a program is controlled
  - What action happens first, second, third, ….
- In movie-style programs (Chaps 1-4) the sequence of actions is determined by the programmer
  - Creating a storyboard design
  - Writing program methods to carry out the designed sequence

Interactive Animations

- In interactive programs, the sequence of actions is determined at runtime, when the user provides **input**
  - Clicks the mouse
  - Presses a key on the keyboard
- Other sources of input are possible

Interactive Games

- In a video game where the user is guiding a spaceship, the sequence of actions …
  - Depends on what direction the user guides the ship
  - How fast the user presses the controls
- Each time the program runs, user input may cause a different sequence of actions
- Control of flow is “in the hands of the user”

You Already Saw Events

- Each time the user provides some sort of input, an event is generated

From Appendix

When spacebar pressed,
Bee turns around

Event Handlers

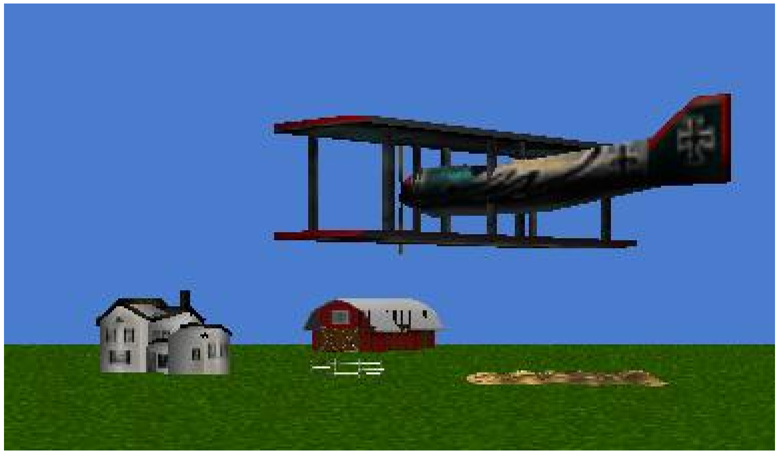
- An event may
  - Trigger a response, or
  - Move objects into positions that create some condition (e.g. a collision) that triggers a response
- An **event handler** is a *method* that is called to carry out the response.
- When an event is linked to an event handler, a **behavior** is created.

How does this effect your program?

- Our goal is to write interactive programs.
- The approach is the same as before, but the difference is now must be concerned with **behaviors**.
  - input from the user (**events**)
  - How objects respond to an event (**event handler methods**)

Example

- Build an air show flight simulator. The pilot (user) uses the biplane controls to perform acrobatic stunts.



- Problem: How do we write program code to provide a guidance system that allows the user to be the pilot?

Solution

- Use keyboard input
  - "F" key to move the biplane forward
  - Spacebar to make the biplane do a barrel roll
  - Note: other keys could be chosen
- Write event handler methods that respond to each key press
- Storyboards (next slide) and DEMO building world

Storyboards

- Since two keys are used, two events are possible – so two storyboards are needed

Event: Spacebar press

Response:

```
Do together
   roll biplane a full revolution
   play biplane engine sound
```

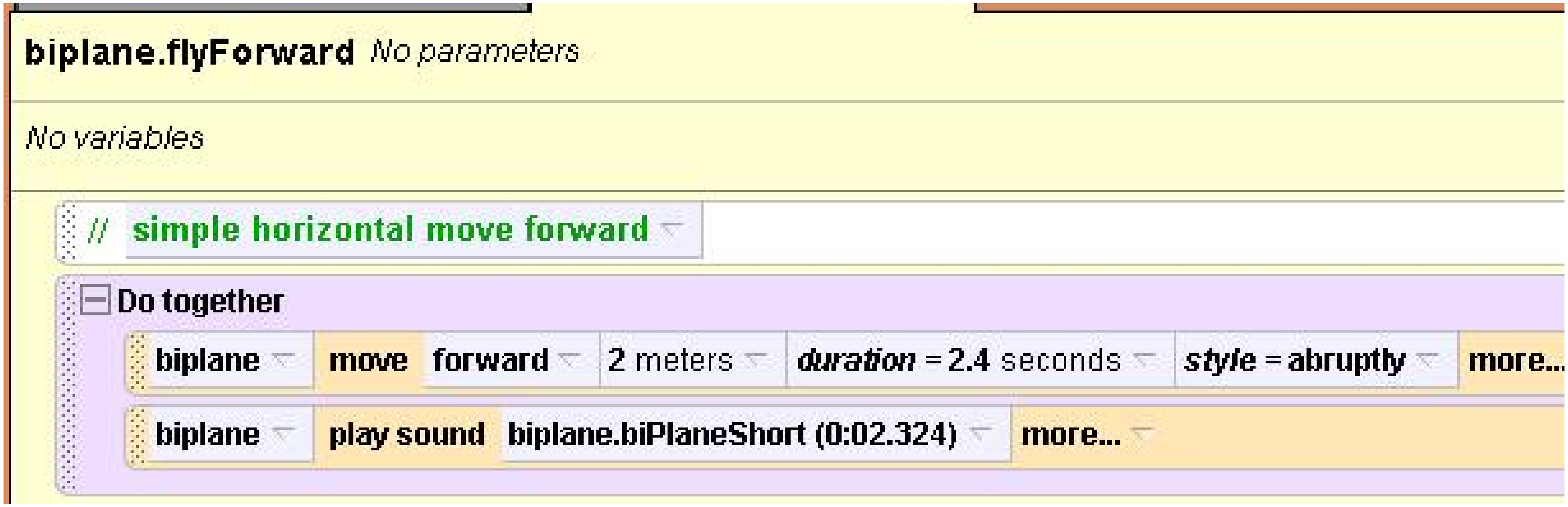
Event: F-key press

Response:

```
Do together
    move biplane forward
    play biplane engine sound
```

- Each storyboard outlines and event handler
  - – Responds to a particular event

biplane.flyForward



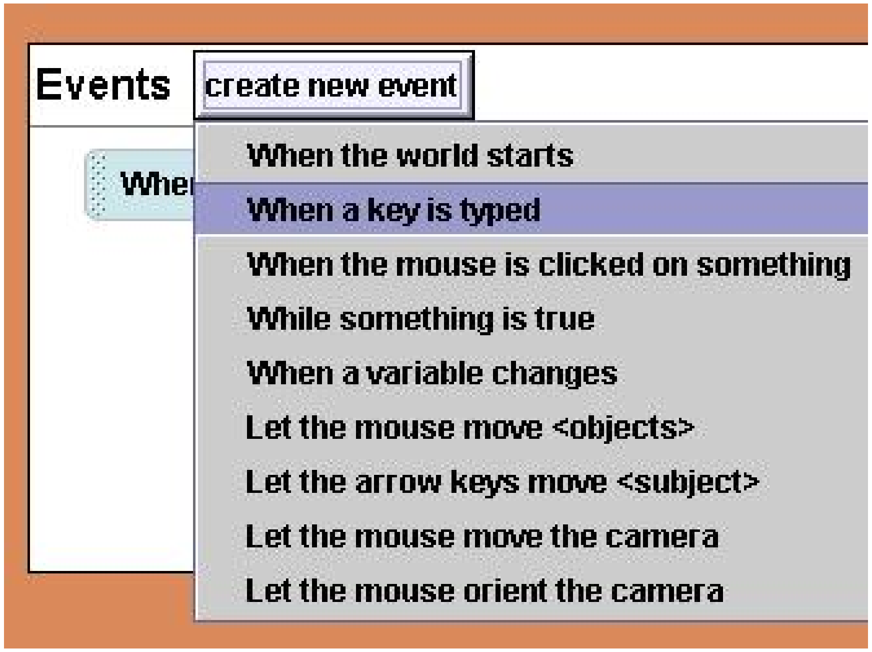
```
biplane.flyForward No parameters

No variables

// simple horizontal move forward
Do together
    biplane  move  forward  2 meters  duration = 2.4 seconds  style = abruptly  more...
    biplane  play sound  biplane.biPlaneShort (0:02.324)  more...
```

- Do not modify the length of the sound
  - – use “as is”
- Coordinate duration of *move* and *play sound*
  - – Match duration of move to duration of sound

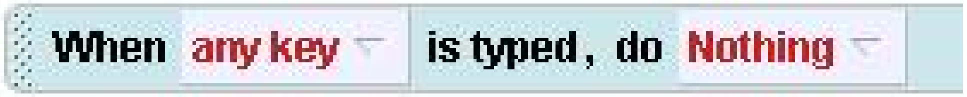
Events Editor - Linking

- Each event handler method must be linked to an event

1) Select “create new event”
Then choose the type of event

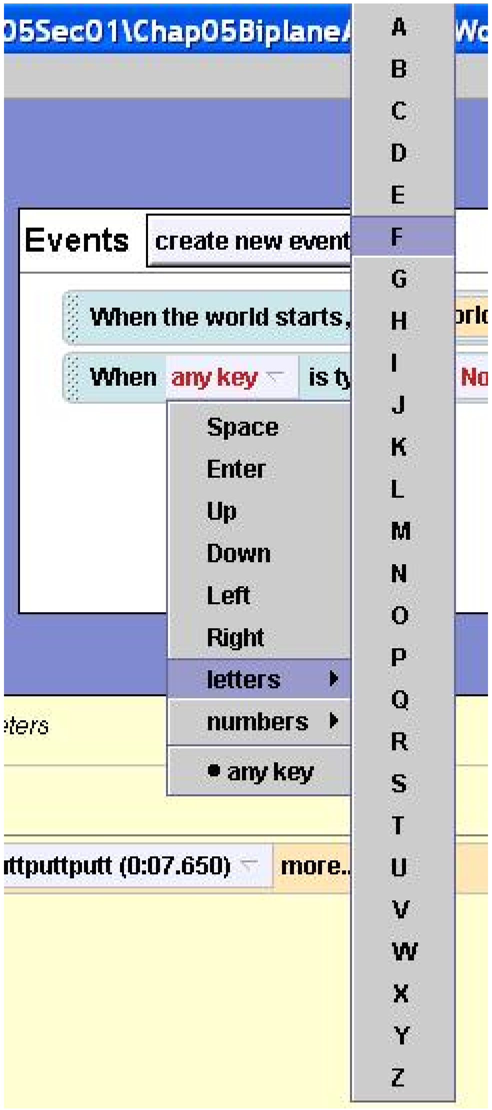

Events
create new event
Whe
When the world starts
When a key is typed
When the mouse is clicked on something
While something is true
When a variable changes
Let the mouse move <objects>
Let the arrow keys move <subject>
Let the mouse move the camera
Let the mouse orient the camera

2) A template linking is created

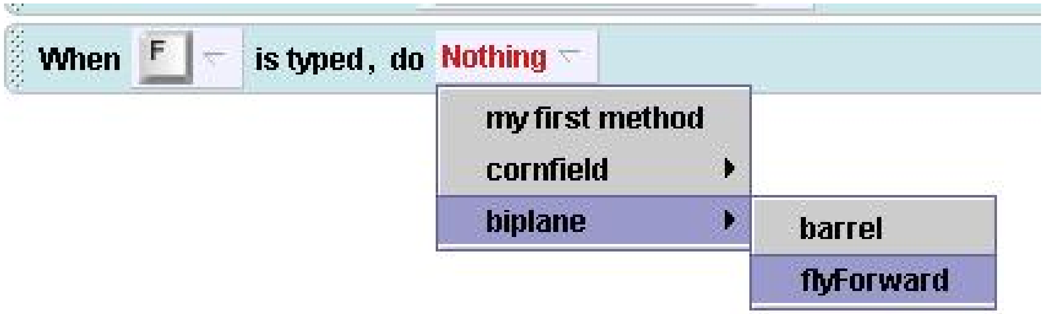


Events Editor – Linking (cont)

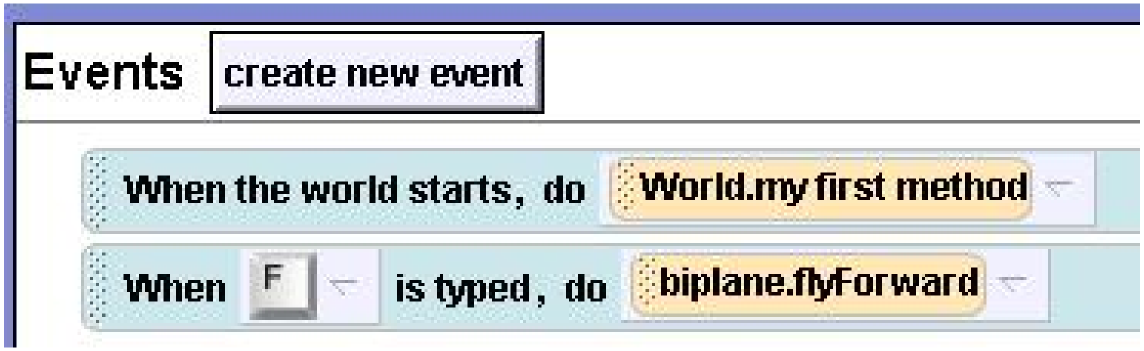
3) Select type of key for event



4) Select event handler method


When F is typed, do Nothing

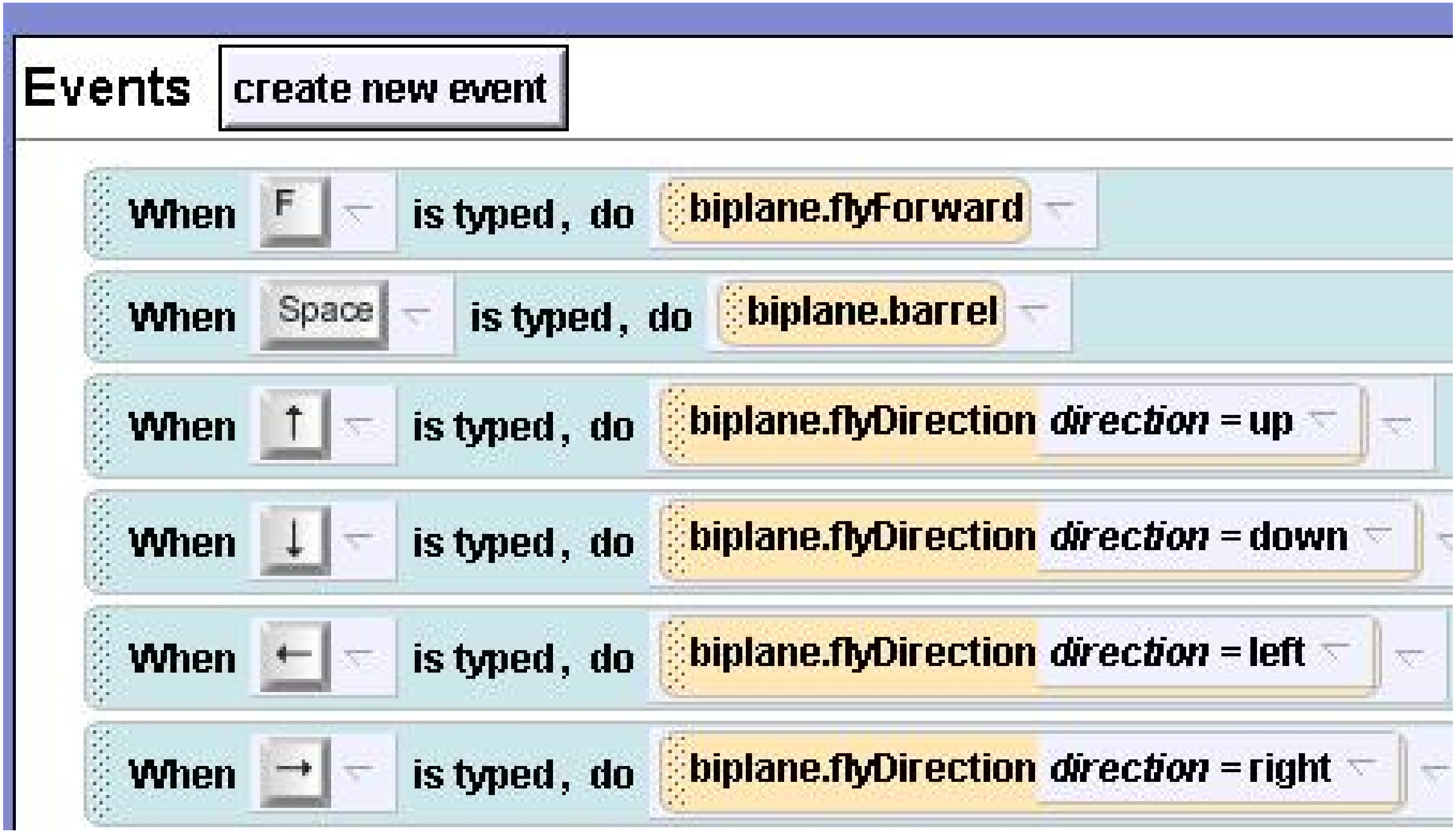
Final result:



When the world starts, do World.my first method

When F is typed, do biplane.flyForward

More Functionality



When F is typed, do biplane.flyForward

When Space is typed, do biplane.barrel

When ↑ is typed, do biplane.flyDirection *direction* = up

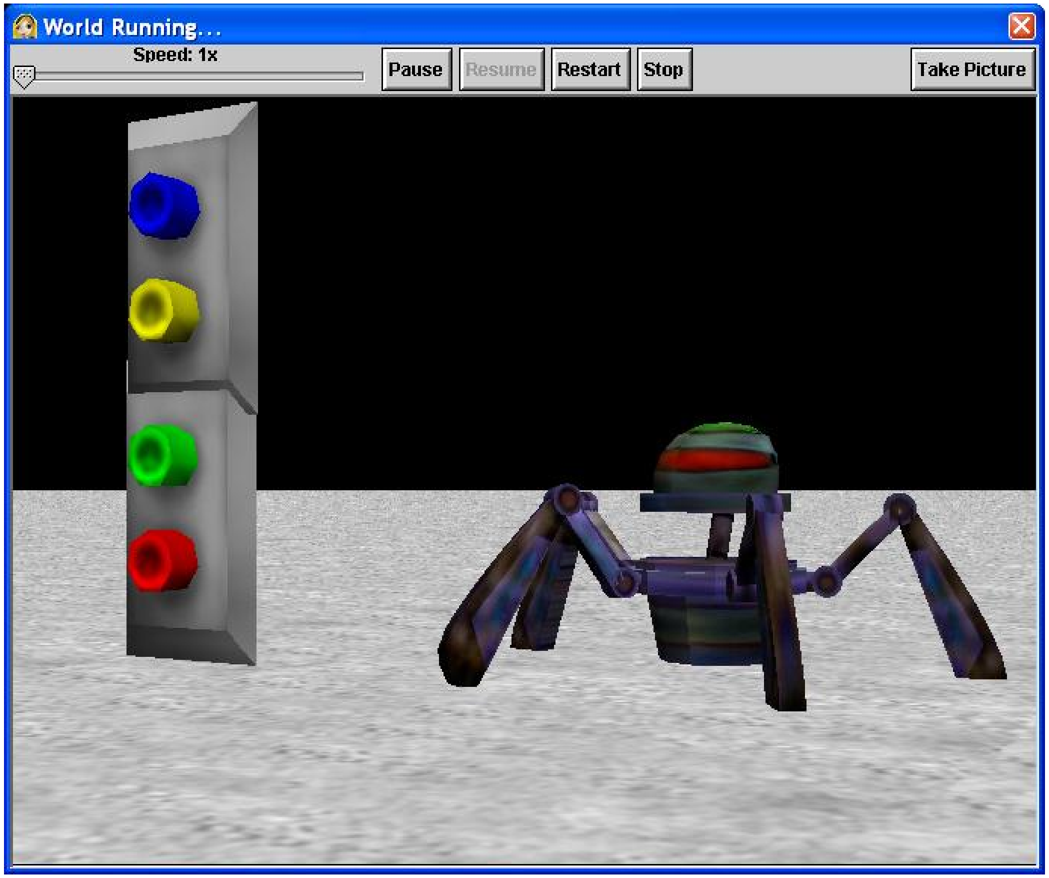
When ↓ is typed, do biplane.flyDirection *direction* = down

When ← is typed, do biplane.flyDirection *direction* = left

When → is typed, do biplane.flyDirection *direction* = right

Classwork today

- Create 4 buttons and a spider robot
- Press green button and spider moves forward
- Press red button and spider moves backward
- Other two buttons?
- Event for instructions


World Running...
Speed: 1x
Pause
Resume
Restart
Stop
Take Picture