

CompSci 94

KeyPressListener, Collision Listeners

November 7, 2024

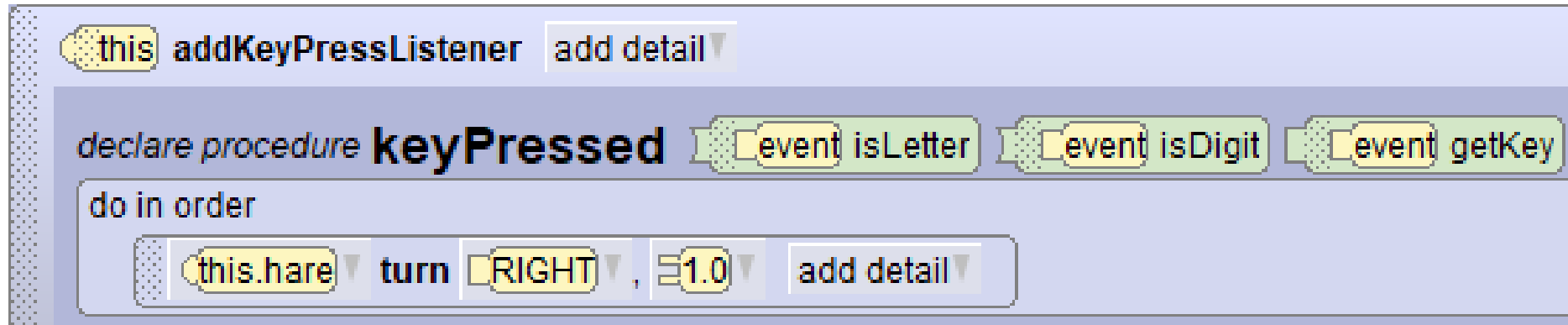


Prof. Susan Rodger

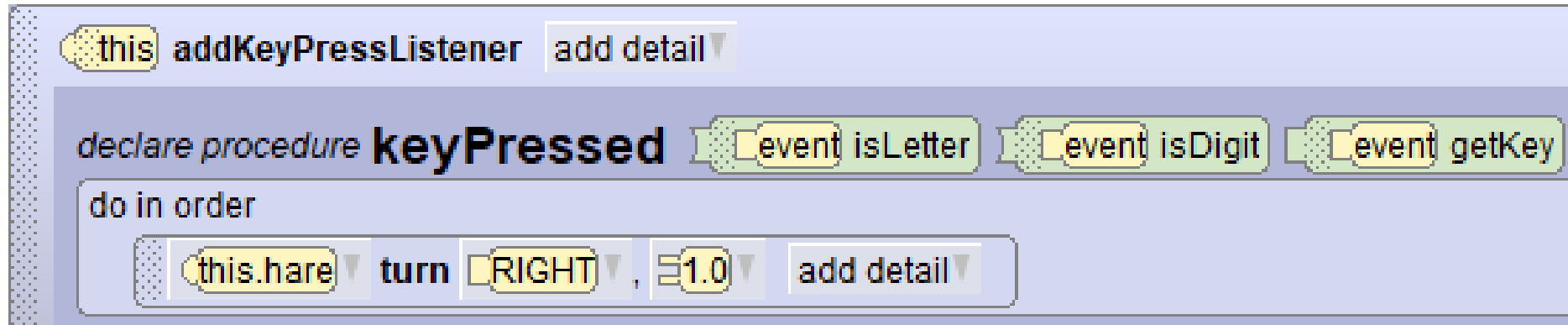
Announcements

- Assignment 5 is due today
- Assignment 6 out, due Nov 21
- Watch videos and online quiz for Thursday
- Exam 3 is November 19
 - Study materials on Nov 19 date on our calendar

Q1: How do I get the hare to turn around?



Q1: How do I get the hare to turn around?



- Press any key and the hare will turn around
- Not a good way to do this. Can't use any other keys for anything else.

Q2: What happens if I press letter A?

If I press the letter T?



Q2: What happens if I press letter A?

If I press the letter T?

- Letter A – pig turns
- Letter T – pig turns, then panda turns



Q3: What happens if press letter A?

If press letter T?



Q3: What happens if press letter A?

If press letter T?

- Letter A – pig turns once
- Letter T – pig turns once



Q4: What does Combine and Fire_Multiple do?

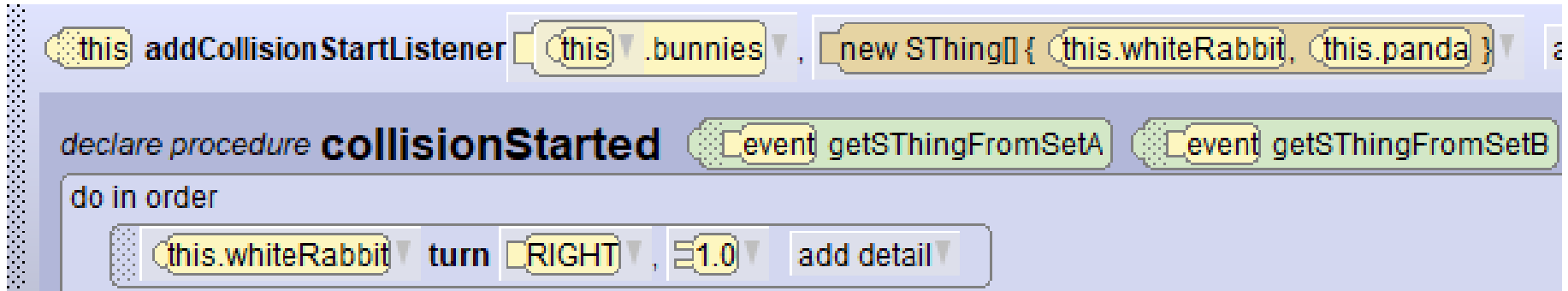


Q4: What does Combine and Fire_Multiple do?

- Hold the key down and the whiteRabbit moves a lot faster until you release the key!



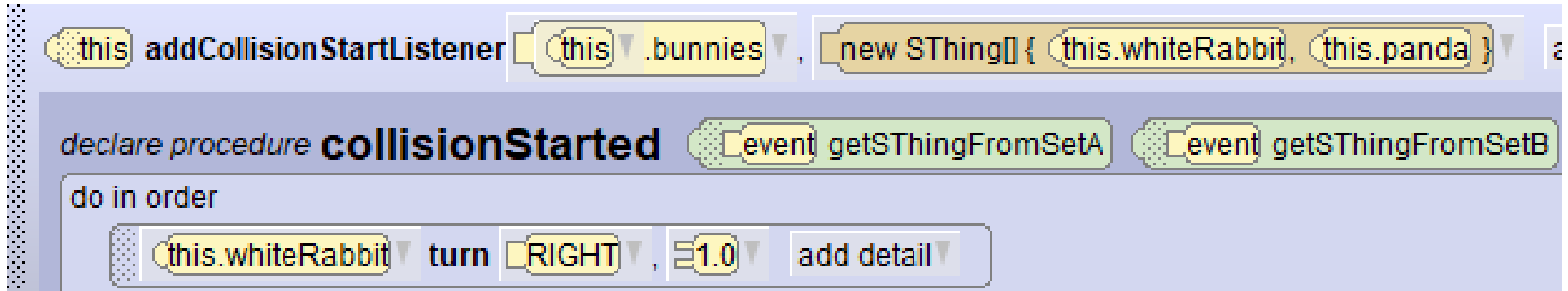
Q5: What happens when ...



- a) panda collides with a bunny?
- b) whiteRabbit collides with a bunny?

Note: bunnies is an array of bunnies

Q5: What happens when ...



a) panda collides with a bunny?

WhiteRabbit (W.R.) turns right

b) whiteRabbit collides with a bunny?

whiteRabbit turns right

Note: bunnies is an array of bunnies

Q6: What happens when

- a) panda collides with a bunny?
- b) whiteRabbit collides with a bunny?
- c) pig collides with a bunny?
- d) whiteRabbit collides with panda?



The image shows a Scratch script for handling collisions. The script starts with a **this** block followed by **addCollisionStartListener**. The listener is a **new SThing[]** block with two arguments: **this.bunnies** and **new SThing[] { this.whiteRabbit, this.panda }**. Below this is a **declare procedure collisionStarted** block. The procedure body begins with **do in order**. Inside, there is an **if** block: **event** **getSThingFromSetB** **==** **this.whiteRabbit** **is true then**. The **then** branch contains **this.whiteRabbit** **say** **"hello"** **add detail**. The **else** branch contains another **if** block: **event** **getSThingFromSetB** **==** **this.panda** **is true then**. This inner **if** block has two branches: **this.panda** **say** **"hello"** **add detail** and **this.pig** **say** **"hello"** **add detail**.

```
this addCollisionStartListener [this.bunnies], new SThing[] { this.whiteRabbit, this.panda }

declare procedure collisionStarted [event] getSThingFromSetA [event] getSThingFromSetB
do in order
  if [event] getSThingFromSetB == [this.whiteRabbit] is true then
    [this.whiteRabbit] say "hello" add detail
  else
    if [event] getSThingFromSetB == [this.panda] is true then
      [this.panda] say "hello" add detail
    else
      [this.pig] say "hello" add detail
```

Q6: What happens when

a) panda collides with a bunny?

Panda says hello

b) whiteRabbit collides with a bunny?

W.R. says hello

c) pig collides with a bunny?

Nothing happens

d) whiteRabbit collides with panda?

Nothing happens



The image shows a Scratch script for collision detection. The script starts with a function call: `this.addCollisionStartListener(this.bunnies, new SThing[] { this.whiteRabbit, this.panda })`. Below this is a procedure declaration: `declare procedure collisionStarted` with two parameters: `event` (type `getSThingFromSetA`) and `getSThingFromSetB` (type `getSThingFromSetB`). The procedure body is enclosed in a `do in order` block. It contains two nested `if` statements. The first `if` statement checks if `event.getSThingFromSetB == this.whiteRabbit` is true. If true, it executes `this.whiteRabbit.say("hello")` and `add detail`. If false, it goes to an `else` block. Inside the `else` block, there is another `if` statement checking if `event.getSThingFromSetB == this.panda` is true. If true, it executes `this.panda.say("hello")` and `add detail`. If false, it goes to another `else` block where it executes `this.pig.say("hello")` and `add detail`.

```
this.addCollisionStartListener(this.bunnies, new SThing[] { this.whiteRabbit, this.panda })

declare procedure collisionStarted
  event getSThingFromSetA  getSThingFromSetB
do in order
  if (event.getSThingFromSetB == this.whiteRabbit) is true then
    this.whiteRabbit.say("hello")
    add detail
  else
    if (event.getSThingFromSetB == this.panda) is true then
      this.panda.say("hello")
      add detail
    else
      this.pig.say("hello")
      add detail
```

Q7: Clicking on an array object

- There is an array of bunnies. When a bunny collides with panda, you want the bunny that collided with the panda to say hello and turn around once.
- Why doesn't this code work?

The image shows a Scratch script for a collision event. The event is triggered by 'this' clicking on a 'new SThing[]' object containing 'this.panda'. The script is attached to 'this.bunnies' and has an 'add detail' block. The procedure 'collisionStarted' is declared with two arguments: 'event getSThingFromSetA' and 'event getSThingFromSetB'. Inside the procedure, there are two blocks: 'this.bunny4 say "hello" add detail' and 'this.bunny4 turn RIGHT, 1.0 add detail'.

```
this addCollisionStartListener new SThing[] { this.panda } , this.bunnies add detail  
  
declare procedure collisionStarted event getSThingFromSetA event getSThingFromSetB  
do in order  
  this.bunny4 say "hello" add detail  
  this.bunny4 turn RIGHT, 1.0 add detail
```

Q7: Clicking on an array object

- There is an array of bunnies. When a bunny collides with panda, you want the bunny that collided with the panda to say hello and turn around once.
- Why doesn't this code work?

Bunny4 says and turns

```
addCollisionStartListener new SThing[] { this.panda }, this.bunnies add detail  
  
declare procedure collisionStarted event getSThingFromSetA event getSThingFromSetB  
do in order  
  this.bunny4 say "hello" add detail  
  this.bunny4 turn RIGHT, 1.0 add detail
```


Q7: Clicking on an array object

- There is an array of bunnies. When a bunny collides with panda, you want the bunny that collided with the panda to say hello and turn around once.
- Can you change the code to this?

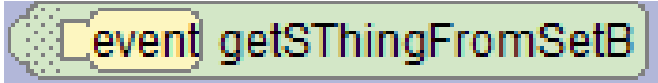
```
addCollisionStartListener new SThing[] { this.panda }, this.bunnies add detail  
  
declare procedure collisionStarted event getSThingFromSetA event getSThingFromSetB  
do in order  
  this.bunny4 say "hello" add detail  
  this.bunny4 turn RIGHT, 1.0 add detail
```

Q7: Clicking on an array object

- There is an array of bunnies. When a bunny collides with panda, you want the bunny that collided with the panda to say hello and turn around once.
- Can you change the code to this? **NO!**

The image shows a snippet of Scratch code. At the top, there is an event listener block: "when green flag clicked" (represented by a dotted circle) followed by "addCollisionStartListener" (a yellow block). The "addCollisionStartListener" block has two arguments: "new SThing[] { this.panda }" (a yellow block) and "this.bunnies" (a yellow block). Below this is a procedure block: "declare procedure collisionStarted" (a blue block). The procedure has two arguments: "event" (a green block) and "getSThingFromSetB" (a green block). The procedure body is enclosed in a "do in order" block (a blue block). Inside the "do in order" block, there are two blocks: "this.bunny4 say 'hello'" (a yellow block) and "this.bunny4 turn RIGHT, 1.0" (a yellow block). Two black arrows point from the "event" argument of the "collisionStarted" procedure to the "this.bunny4" argument of both blocks in the "do in order" block.

Why not?

- This code: 
 - Is an Sthing so you CANNOT drop it over a type bunny
- Instead, you have to look through the bunny array and compare each bunny with with an Sthing. When you find the bunny that was clicked on, then you just refer to that bunny

Find bunny clicked on in array

- Write a loop to iterate through the bunny array, for each bunny in the array, check to see if it is the item clicked on.



The image shows a Scratch script for collision detection. At the top, a 'this' block is connected to 'addCollisionStartListener', which is followed by a 'new SThing[]' block containing '{ this.panda }'. This is then connected to a 'this.bunnies' block, which is followed by an 'add detail' block. Below this, a 'declare procedure collisionStarted' block is shown, which contains an 'event' block connected to 'getSThingFromSetA' and another 'event' block connected to 'getSThingFromSetB'. The procedure body starts with 'do in order', followed by a 'for each' loop. The loop iterates over 'Bunny' objects, with 'someBunny' being the current item in 'this.bunnies'. Inside the loop, there is an 'if' statement: 'if [event getSThingFromSetB == someBunny] is true then'. If true, the code blocks 'this.bunny4 say "hello"', 'add detail', 'this.bunny4 turn RIGHT', and '1.0' are executed, followed by 'add detail'. If false, a dashed box labeled 'drop statement here' is shown. The loop is closed with a 'loop' block.

```

this addCollisionStartListener new SThing[] { this.panda }, this.bunnies add detail

declare procedure collisionStarted
  event getSThingFromSetA event getSThingFromSetB
  do in order
    for each Bunny someBunny in this.bunnies
      if [event getSThingFromSetB == someBunny] is true then
        this.bunny4 say "hello" add detail
        this.bunny4 turn RIGHT, 1.0 add detail
      else
        drop statement here
    loop

```

Find bunny clicked on in array

- Write a loop to iterate through the bunny array, for each bunny in the array, check to see if it is the item clicked on.

The image shows a Scratch code editor with the following code:

```
addCollisionStartListener new SThing [ { this.panda } ], [ this.bunnies ] add detail  
  
declare procedure collisionStarted [ event ] getSThingFromSetA [ event ] getSThingFromSetB  
do in order  
  for each Bunny [ someBunny ] in [ this.bunnies ]  
    if [ event ] getSThingFromSetB == [ someBunny ] is true then  
      [ this.bunny4 ] say [ "hello" ] add detail  
      [ this.bunny4 ] turn [ RIGHT ], [ 1.0 ] add detail  
    else  
      drop statement here  
  loop
```

A green box highlights the `if` statement and its contents. The word **COMPARE** is written in large green letters to the right of the code.

Find bunny clicked on in array

- Write a loop to iterate through the bunny array, for each bunny in the array, check to see if it is the item clicked on.



Find bunny clicked on in array

- Write a loop to iterate through the bunny array, for each bunny in the array, check to see if it is the item clicked on.



Class Today

- A game with collisions

