

CompSci 94

KeyPressListener, Collision Listeners

November 4, 2021



Prof. Susan Rodger

Announcements

- Assignment 5 is due Thursday, November 11
- Watch videos and online quiz for Tuesday
- Exam 2 is November 16

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

this addKeyPressListener add detail

declare procedure **keyPressed** event isLetter event isDigit event getKey

do in order

this.hare turn RIGHT, 1.0 add detail

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

```
this addKeyPressListener add detail ▼  
  
declare procedure keyPressed event isLetter event isDigit event getKey  
do in order  
  this.hare turn RIGHT, 1.0 add detail ▼
```

- 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?

The image shows a Scratch code editor with a procedure named `keyPressed`. The procedure is triggered by the event `addKeyPressListener` with the detail `add detail`. The procedure is defined as follows:

```
declare procedure keyPressed event isLetter event isDigit
do in order
  if event isLetter is true then
    this.pig turn RIGHT, 1.0 add detail
  else
    drop statement here
  if event isKey T is true then
    this.panda turn RIGHT, 1.0 add detail
  else
    drop statement here
```

The code is written in a block-based style. The procedure is triggered by the event `addKeyPressListener` with the detail `add detail`. The procedure is defined as follows:

- `declare procedure keyPressed` with parameters `event isLetter` and `event isDigit`.
- `do in order` block containing:
 - `if event isLetter is true then` block:
 - `this.pig turn RIGHT, 1.0 add detail` block.
 - `else` block with a dashed box labeled `drop statement here`.
 - `if event isKey T is true then` block:
 - `this.panda turn RIGHT, 1.0 add detail` block.
 - `else` block with a dashed box labeled `drop statement here`.

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

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

```
this addKeyPressListener add detail  
  
declare procedure keyPressed event isLetter event isDigit  
do in order  
  if event isLetter is true then  
    this.pig turn RIGHT, 1.0 add detail  
  else  
    drop statement here  
  if event isKey T is true then  
    this.panda turn RIGHT, 1.0 add detail  
  else  
    drop statement here
```

The code defines a procedure named `keyPressed` that takes two arguments: `event isLetter` and `event isDigit`. The procedure performs the following actions in order:

- Check if `event isLetter` is true. If true, turn `this.pig` `RIGHT` by `1.0` degrees and add a detail.
- Check if `event isKey T` is true. If true, turn `this.panda` `RIGHT` by `1.0` degrees and add a detail.

Q3: What happens if press letter A? If press letter T?

The image shows a Scratch code editor window with a procedure named `keyPressed`. The procedure is triggered by the `addKeyPressListener` block. The code inside the procedure is as follows:

```
declare procedure keyPressed
do in order
  if event isLetter is true then
    this.pig turn RIGHT, 1.0 add detail
  else
    if event isKey T is true then
      this.panda turn RIGHT, 1.0 add detail
    else
      drop statement here
```

The code is written in a block-based style. The `keyPressed` procedure is defined with a `do in order` block. Inside this block, there is an `if` statement. The first `if` statement checks if the `event` is a letter. If true, it turns `this.pig` `RIGHT` by `1.0` degrees and adds a detail. If false, it goes to an `else` block. Inside the `else` block, there is another `if` statement that checks if the `event` is the key `T`. If true, it turns `this.panda` `RIGHT` by `1.0` degrees and adds a detail. If false, it goes to a `drop statement here` block.

Q3: What happens if press letter A? If press letter T?

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

```
addKeyPressListener add detail
declare procedure keyPressed
  event isLetter event isDigit
do in order
  if event isLetter is true then
    this.pig turn RIGHT, 1.0 add detail
  else
    if event isKey T is true then
      this.panda turn RIGHT, 1.0 add detail
    else
      drop statement here
```

The image shows a Scratch code editor snippet. At the top, there is a yellow button labeled 'this' followed by the text 'addKeyPressListener' and a dropdown menu 'add detail'. Below this is a procedure declaration: 'declare procedure keyPressed' with two input fields: 'event isLetter' and 'event isDigit'. The procedure body starts with 'do in order'. Inside, there is an 'if' block: 'if event isLetter is true then'. This block contains a 'turn' block: 'this.pig turn RIGHT, 1.0 add detail'. Below the 'if' block is an 'else' block. Inside the 'else' block, there is another 'if' block: 'if event isKey T is true then'. This block contains a 'turn' block: 'this.panda turn RIGHT, 1.0 add detail'. Below this 'if' block is another 'else' block containing a dashed box with the text 'drop statement here'.

Q4: What does Combine and Fire_Multiple do?

```
this addKeyPressListener, multipleEventPolicy COMBINE, heldKeyPolicy FIRE_MULTIPLE
```

```
declare procedure keyPressed event isLetter event isDigit event getKey
```

```
do in order
```

```
if event isKey RIGHT is true then
```

```
this.whiteRabbit move RIGHT, 0.25 add detail
```

```
else
```

```
if event isKey UP is true then
```

```
this.whiteRabbit move FORWARD, 0.25 add detail
```

```
else
```

```
drop statement here
```

Q4: What does Combine and Fire_Multiple do?

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

```
this addKeyPressListener, multipleEventPolicy COMBINE, heldKeyPolicy FIRE_MULTIPLE  
  
declare procedure keyPressed event isLetter event isDigit event getKey  
do in order  
  if event isKey RIGHT is true then  
    this.whiteRabbit move RIGHT, 0.25 add detail  
  else  
    if event isKey UP is true then  
      this.whiteRabbit move FORWARD, 0.25 add detail  
    else  
      drop statement here
```

Q5: What happens when ...

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

a) panda collides with a bunny?

b) whiteRabbit collides with a bunny?

Note: bunnies is an array of bunnies

Q5: What happens when ...

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

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?

```
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

```
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
```

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?

```
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

```
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?

The image shows a snippet of Scratch code. At the top, there is an event listener: `this addCollisionStartListener` with a dropdown menu containing `new SThing[] { this.panda }` and another dropdown menu containing `this .bunnies`. Below this is a procedure declaration: `declare procedure collisionStarted` with two event dropdown menus: `event getSThingFromSetA` and `event getSThingFromSetB`. The procedure body is enclosed in a `do in order` block and contains two actions: `this.bunny4 say "hello" add detail` and `this.bunny4 turn RIGHT, 1.0 add detail`. Two black arrows point from the `event getSThingFromSetA` dropdown to the `this.bunny4` object in the first action, and from the `event getSThingFromSetB` dropdown to the `this.bunny4` object in the second action.

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 a block: `this addCollisionStartListener new SThing[] { this.panda }, this.bunnies add detail`. Below this is a procedure declaration: `declare procedure collisionStarted event getSThingFromSetA event getSThingFromSetB`. The procedure body contains two blocks: `this.bunny4 say "hello" add detail` and `this.bunny4 turn RIGHT, 1.0 add detail`. Two black arrows originate from the `event getSThingFromSetA` and `event getSThingFromSetB` blocks and point to the `this.bunny4` object in the first block of the procedure, indicating that the event handlers are incorrectly targeting a specific bunny instead of the one that triggered the collision.

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 code editor window 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
```

The code defines a procedure named `collisionStarted` that takes two event objects as arguments. It enters a `do in order` block containing a `for each` loop over the `this.bunnies` array. Inside the loop, an `if` statement checks if the event's `getSThingFromSetB` property matches the current `someBunny`. If true, it triggers `say "hello"` and `turn RIGHT, 1.0` on `this.bunny4`. An `else` block contains a dashed box labeled "drop statement here". A `loop` bracket is placed at the bottom of the `for each` 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 window. At the top, there is a line of code: `this addCollisionStartListener new SThing[] { this.panda } , this .bunnies add detail`. Below this, a procedure is declared: `declare procedure collisionStarted event getSThingFromSetA event getSThingFromSetB`. The procedure body starts with `do in order`. Inside, there is a `for each Bunny someBunny in this .bunnies` loop. Within this loop, an `if event getSThingFromSetB == someBunny is true then` block is highlighted with a green border. This block contains two actions: `this.bunny4 say "hello" add detail` and `this.bunny4 turn RIGHT , 1.0 add detail`. An `else` block with a dashed border contains the text `drop statement here`. A `loop` bracket is visible at the bottom left of the loop structure. The word **COMPARE** is written in large green letters on the right side of the code editor.

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.

```
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  
      someBunny say "hello" add detail  
      someBunny turn RIGHT, 1.0 add detail  
    else  
      drop statement here  
  loop
```

FINAL CODE

Class Today

- A game with collisions

