

# 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** ▾

# Q2: What happens if I press letter A? If I press the letter T?

The image shows a Scratch code editor 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`. Inside the procedure, there are two conditional blocks. The first block checks if `event isLetter` is true; if so, it turns `this.pig` right by 1.0 degrees and adds a detail. The second block checks if `event isKey T` is true; if so, it turns `this.panda` right by 1.0 degrees and adds a detail. Both blocks have a dashed box labeled "drop statement here" as an alternative action.

# 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 `if` blocks are nested. The first `if` block checks if the event is a letter. If true, it turns the pig 1.0 degrees to the right. The `else` block checks if the event is the key 'T'. If true, it turns the panda 1.0 degrees to the right. The `else` block for the second `if` is currently empty, indicated by a dashed box labeled "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
```

# 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

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
```



# 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
```

# Class Today

- A game with collisions

