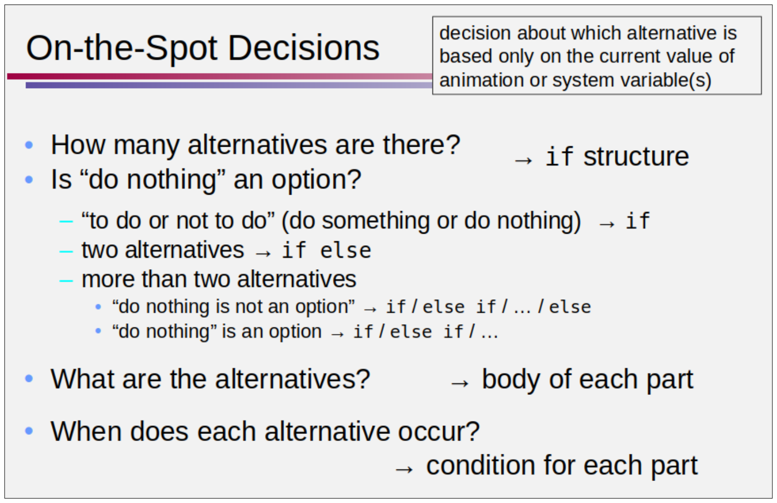
Names: ____

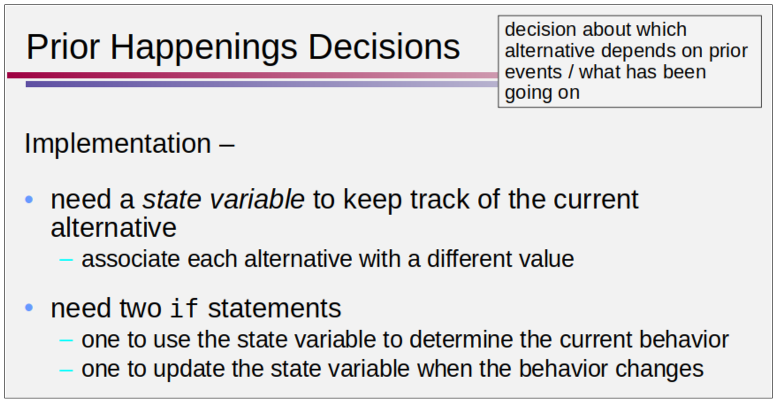
Mapping the Conditionals Questions to Code



On-the-Spot Decisions

decision about which alternative is based only on the current value of animation or system variable(s)

- How many alternatives are there? → `if` structure
- Is "do nothing" an option?
  - "to do or not to do" (do something or do nothing) → `if`
  - two alternatives → `if else`
  - more than two alternatives
    - "do nothing is not an option" → `if` / `else if` / … / `else`
    - "do nothing" is an option → `if` / `else if` / …
- What are the alternatives? → body of each part
- When does each alternative occur? → condition for each part

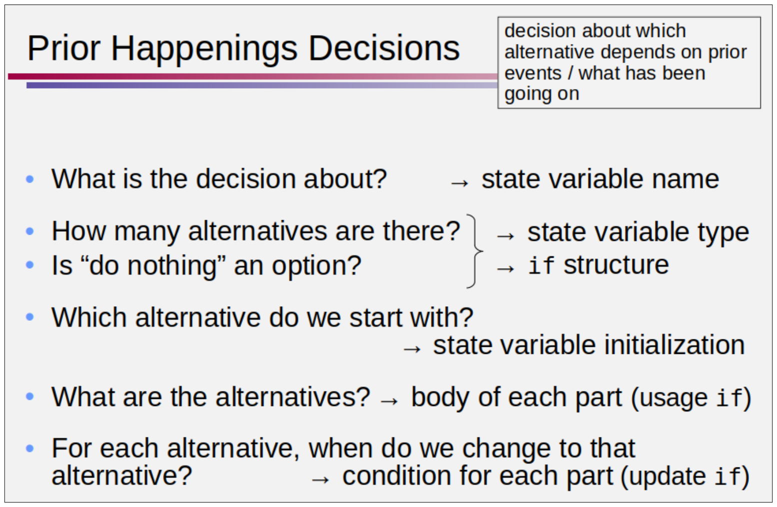


Prior Happenings Decisions

decision about which alternative depends on prior events / what has been going on

Implementation –

- need a *state variable* to keep track of the current alternative
  - associate each alternative with a different value
- need two `if` statements
  - one to use the state variable to determine the current behavior
  - one to update the state variable when the behavior changes

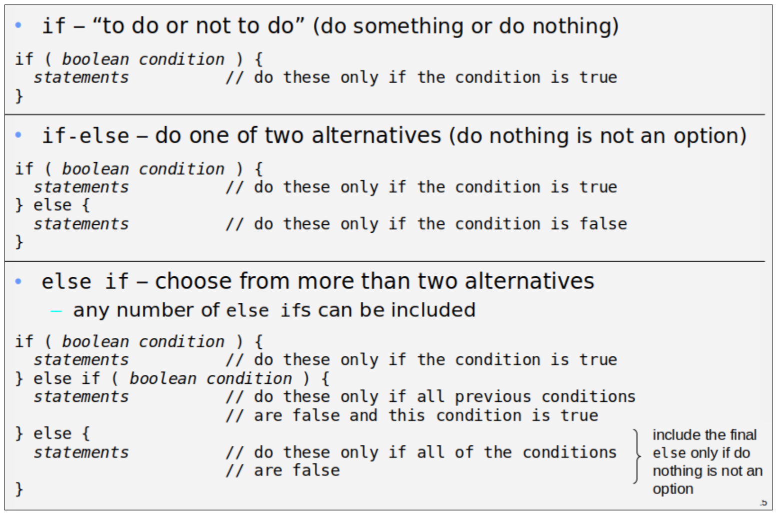


Prior Happenings Decisions

decision about which alternative depends on prior events / what has been going on

- What is the decision about? → state variable name
- How many alternatives are there? → state variable type
- Is "do nothing" an option? → `if` structure
- Which alternative do we start with? → state variable initialization
- What are the alternatives? → body of each part (usage `if`)
- For each alternative, when do we change to that alternative? → condition for each part (update `if`)

if Statements – Syntax and Structural Patterns



- `if` – "to do or not to do" (do something or do nothing)

```
if ( boolean condition ) {
  statements          // do these only if the condition is true
}
```

- `if-else` – do one of two alternatives (do nothing is not an option)

```
if ( boolean condition ) {
  statements          // do these only if the condition is true
} else {
  statements          // do these only if the condition is false
}
```

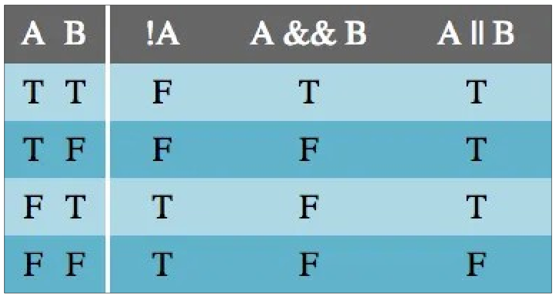
- `else if` – choose from more than two alternatives
  - any number of `else if`s can be included

```
if ( boolean condition ) {
  statements          // do these only if the condition is true
} else if ( boolean condition ) {
  statements          // do these only if all previous conditions
                      // are false and this condition is true
} else {
  statements          // do these only if all of the conditions
                      // are false
}
```

include the final `else` only if do nothing is not an option

boolean condition is an expression that evaluates to true or false – typically a comparison involving ==, !=, <, <=, >, >= or ! or two or more conditions joined by && or ||

statements are any legal Processing statements e.g. drawing commands, function calls, animation variable updates, conditionals, …



A	B	!A	A && B	A || B
T	T	F	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	F

At the End of Class

Hand in whatever you have done during class, even if a sketch is incomplete.

- Make sure each sketch is named as directed and has a comment with the names of your group. Also be sure to save your sketches! (This should be in your sketchbook **~/cs120/sketchbook**)
- Copy the entire directory for each sketch (not only the `.pde` file) into your handin directory (**/classes/cs120/handin/*username***). You only need to hand in one copy for the group.

Exercises

These are the same sketches as from Monday's in class exercises, but now you are writing code. *Make sure that you've answered the conditionals questions before writing code!* Refer to the posted solutions and to your own group's handout.

1. Create a sketch called **sketch_251008a** which contains a 100x100 rectangle centered in the drawing window. The rectangle should be outlined in black, and should be filled red when the mouse is over it and blue otherwise.

2. Create a sketch called **sketch_251008b** which contains a 100x100 rectangle centered in the drawing window. The rectangle should grow in width when the mouse is over the rectangle, and should reset to 100 when it reaches the edge of the window. (The rectangle should remain centered in the window.)

3. Create a sketch called **sketch_251008c** which contains an ellipse which grows steadily. Clicking the mouse should pause and unpause the animation so that sometimes the ellipse grows and sometimes it doesn't.

4. Create a sketch called **sketch_251008d** which contains a rectangle whose width grows until it reaches the edges of the window, then its height grows. Once the rectangle reaches the top/bottom of the window, both width and height reset to 0 and the animation repeats.

If you have time –

5. Create a sketch called **sketch_251008e** which contains a 100x100 rectangle centered in the drawing window. The rectangle should grow in width when the mouse is moved to the right, shrink in width when the mouse is moved to the left, and not change when the mouse stays still (or moves only vertically). (Hint: in addition to system variables `mouseX` and `mouseY` for the mouse's current position, there are also system variables `pmouseX` and `pmouseY` for the mouse's previous position.)

6. Create a sketch called **sketch_251008f** which contains a 100x100 rectangle centered at a random position in the drawing window. Clicking the mouse inside the rectangle should cause it to move to a new random position.

7. Create a sketch called **sketch_251008g** which contains a rectangle whose width grows until it reaches the edges of the window, then its height grows. Once the rectangle reaches the top/bottom of the window, both width and height reset to 0 and the animation repeats. This repeats three times before stopping.