Mouse and Keyboard

Today’s lab will cover interface design with the mouse and keyboard.

# Task 1 Mouse Events

Tkinter provides a list of mouse events, including mouse button pressed, mouse move, mouse button released et al. You can find more via: <https://www.python-course.eu/tkinter_events_binds.php>

Create a project “Lab-Mouse Interaction” in PyCharm.

Download mouse\_events\_window.py on LMS and copy it to the project folder. Open the file in PyCharm.

1. Mouse Button Pressed
2. In “def \_\_init\_\_(self, window\_width, window\_height, master=None):”, add the code below. It means that when you click left button of the mouse on the red canvas, self.mouse\_left\_button\_press() will be called. The left mouse button is defined by the event <Button-1> (the same as <ButtonPress-1>), the middle button by <Button-2>, and the right mouse button by <Button-3>.

self.canvas.bind("<ButtonPress-1>", self.mouse\_left\_button\_press)

1. Create a function named self.mouse\_left\_button\_press(self, event). It will print the message on the console. Event.x and event.y represent the coordinate point of the cursor (origin is the top left corner of the canvas) when the button is pressed. Note that the function is only called when the cursor is on the canvas. Please think why.

def mouse\_left\_button\_press(self, event):  
 print("Mouse Left Button Pressed on the Canvas", event.x, " ", event.y)

1. Please complete the middle button pressed, and the right mouse button pressed.
2. Mouse Motion
3. In “def \_\_init\_\_(self, window\_width, window\_height, master=None):”, add the code below. It means that when the mouse is moved on canvas, self.mouse\_move() will be called.

self.canvas.bind("<Motion>", self.mouse\_move)

1. To specify mouse move with the left, middle or right mouse button being held down, use <B1-Motion>, <B2-Motion> and <B3-Motion> respectively. For example,

self.canvas.bind("<B1-Motion>", self.mouse\_move\_left\_button\_press)

1. Create functions named def mouse\_move(self, event) and def mouse\_move\_left\_button\_press(self, event). It will print the message on the console. Event.x and event.y represent the coordinate point of the cursor (origin is the top left corner of the canvas).

def mouse\_move(self, event):  
 print("Mouse Move", event.x, " ", event.y)  
  
def mouse\_move\_left\_button\_press(self, event):  
 print("Mouse Move with Left Button Pressed", event.x, " ", event.y)

1. Please complete mouse motion with the middle button pressed, and the right mouse button pressed.
2. Mouse Button Released
3. In “def \_\_init\_\_(self, window\_width, window\_height, master=None):”, add the code below. It means that when you release the left button of the mouse on the red canvas, self.mouse\_left\_button\_release() will be called. The left mouse button is defined by the event <ButtonRelease-1>, the middle button by <ButtonRelease-2>, and the right mouse button by <ButtonRelease-3>.

self.canvas.bind("<ButtonRelease-1>", self.mouse\_left\_button\_release)

1. Create a function named self.mouse\_left\_button\_release(self, event). It will print the message on the console. Event.x and event.y represent the coordinate point of the cursor (origin is the top left corner of the canvas) when the left button is released.

def mouse\_left\_button\_release(self, event):  
 print("Mouse Left Button Released", event.x, " ", event.y)

1. Please complete the middle button released, and the right mouse button released.
2. Mouse Double Click
3. In “def \_\_init\_\_(self, window\_width, window\_height, master=None):”, add the code below. It means that when you double click the left button of the mouse on the red canvas, self.mouse\_left\_button\_double\_click() will be called. The left mouse button is defined by the event <Double-Button-1>, the middle button by <Double-Button-2>, and the right mouse button by <Double-Button-3>.

self.canvas.bind("<Double-Button-1>", self.mouse\_left\_button\_double\_click)

1. Create a function named self. mouse\_left\_button\_double\_click(self, event). It will print the message on the console. Event.x and event.y represent the coordinate point of the cursor (origin is the top left corner of the canvas) when double clicking the left button.

def mouse\_left\_button\_double\_click(self, event):  
 print("Mouse Left Button Double Click", event.x, " ", event.y)

1. Please complete the middle button double clicked, and the right mouse button double clicked.
2. Enter
3. In “def \_\_init\_\_(self, window\_width, window\_height, master=None):”, add the code below. It means that when the mouse pointer enters the red canvas, self.mouse\_pointer\_enter() will be called.

self.canvas.bind("<Enter>", self.mouse\_pointer\_enter)

1. Create a function named self.mouse\_pointer\_enter(self, event). It will print the message on the console. Event.x and event.y represent the coordinate point of the cursor (origin is the top left corner of the canvas) when the pointer enters the canvas.

def mouse\_pointer\_enter(self, event):  
 print("Mouse Pointer Enter", event.x, " ", event.y)

1. Leave
2. In “def \_\_init\_\_(self, window\_width, window\_height, master=None):”, add the code below. It means that when the mouse pointer leaves the red canvas, self.mouse\_pointer\_leave() will be called.

self.canvas.bind("<Leave>", self.mouse\_pointer\_leave)

1. Create a function named self.mouse\_pointer\_leave(self, event). It will print the message on the console. Event.x and event.y represent the coordinate point of the cursor (origin is the top left corner of the canvas) when the pointer leaves the canvas.

def mouse\_pointer\_leave(self, event):  
 print("Mouse Pointer Leave", event.x, " ", event.y)

# Task 2 Gesture Keyboards

In this lab, we will implement an on-screen gesture keyboard that is used with the mouse.

Create a project “Lab-Keyboard Interaction” in PyCharm.

Download GestureKeyboard.zip on LMS and unzip the files to the project: “Lab-Keyboard Interaction”. Open the project in PyCharm.

Run the code and you will see a keyboard application. The application window has three parts. The top is used to show input words, the middle is used to list word candidates, and the bottom is the keyboard.

There are four files in the project folder.

1. text\_entry\_window.py: the \_\_init\_\_() sets the interface layout of the application and also initialises the variables. mouse\_left\_button\_press(), mouse\_left\_button\_release(), mouse\_move\_left\_button\_down() respond to mouse events.
2. template.py: In the class WordTemplates, you can define template words in self.word\_list. You can call def set\_templates(self) to set word-gesture templates (self.templates). self.templates stores the template word and the key position (the centre point of the key) for each letter of the word (i.e., point coordinates for the word-gesture template).
3. recognizer.py: the application needs to record the trajectory points when users draw a word gesture on the keyboard, and then check which template the drawn word gesture matches based on the recognizer. You can provide word candidates according to the recognition results (i.e. sort the distances between the drawn gesture and the templates, which are provided by the recognizer). As illustrated in Fig.1, when participants draw a stroke gesture along the key “T”, “H” and “E”, four word candidates will be shown. And users can select one candidate by clicking it.
4. keyboard\_design.py: this module includes the keyboard layout design and key events.

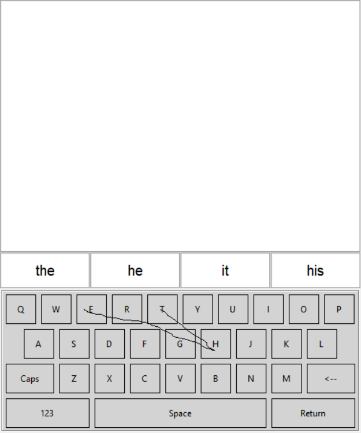
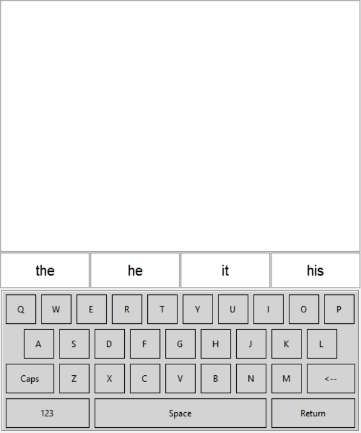
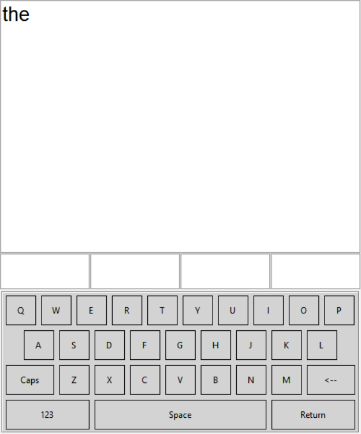
  

Fig. 1: illustrations of gesture keyboard. Left: drawing a word-gesture. Middle: four word candidates. Right: selecting “the”.

**Task**: implement the function of double clicking a key on the keyboard to input the character in uppercase. Hints: you need to add a double click mouse event to canvas\_keyboard. In the event, get the letter of the clicked key: self.keyboard.get\_key\_pressed(). Then use self.label\_word\_candidates[0].config(text=?) to change the content of the first word candidate label.

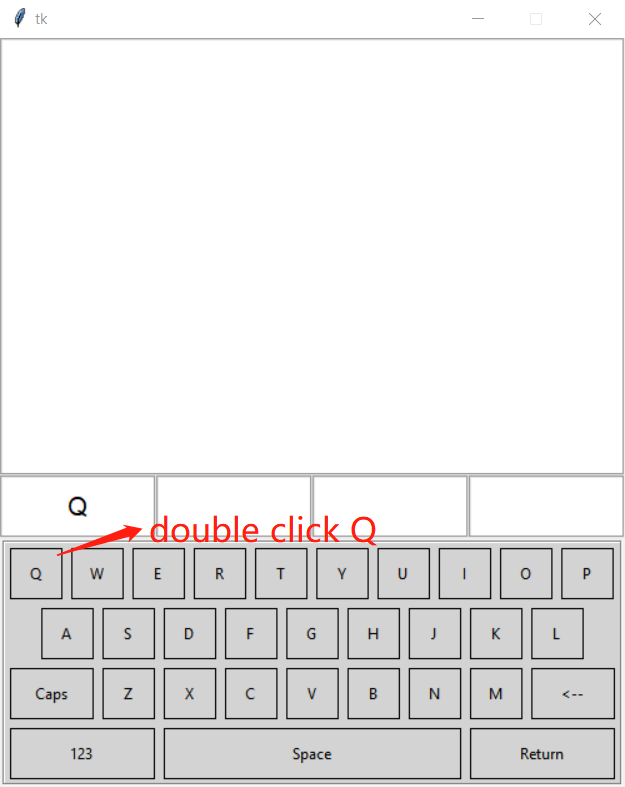


Fig. 2: illustration of double clicking the Q key

Please Complete Student Feedback on LMS. Thank You.