Advanced graphical input and output

Abhiram Ranade

December 30, 2021

1 Input

Simplecpp allows additional ways for users to interact graphically, in addition to `getClick`. Your program can either wait for the user to drag the mouse or press a key or a mouse button, or it can check whether such an event happened recently.

For this it must first create an event object, of class `XEvent`. This can hold information about events.

Waiting for an event: The call `nextEvent(event)` will wait for an event to happen, then record information about the event in `event` and then return. The `event` object is passed by reference.

Checking for events: The call `checkEvent(event)` checks if any event happened (after the last call to `nextEvent` or `checkEvent`) ; if so the information is recoreded in `event`, and the function returns immediately.

Additional functions are provided to extract information about the event that happened.

- The call `mouseButtonPressEvent(event)` returns `true` if the event recorded in `event` is caused by a mouse button press.
- The call `mouseButtonReleaseEvent(event)` returns `true` if the event recorded in `event` is caused by a mouse button release.
- The call `mouseDragEvent(event)` returns `true` if the event recorded `event` is caused by a mouse drag, i.e. the user dragged the mouse after pressing a mouse button.
- The call `keyPressEvent(event)` returns `true` if event recorded in `event` was the pressing of a key of the keyboard.

You can get more information about the event using the members `event.xbutton.button`, which equals an integer denoting which button was pressed/released, and `event.xbutton.x` and `event.xbutton.y` which give the coordinates of the mouse at the time the of the operation.

Here is an example.

```
XEvent event;
nextEvent(event);
if(mouseButtonPressEvent(event)){
  cout <<"Mouse button "<< event.xbutton.button
       <<" pressed, at position ("<< event.xbutton.x <<
       <<", "<< event.xbutton.y << endl;
}
```

This code will cause the program to wait until some event happens, and then if the event was the pressing of some mouse button, it will print which button was pressed (i.e. 1, 2 or 3 to denote left, middle, right) and at what canvas coordinates.

Likewise, the call `charFromEvent(event)` returns the `char` denoting the key that was pressed. The members `event.xkey.x` and `event.xkey.y` respectively give the coordinates of the mouse position at which the key was pressed.

Key press events from some keyboards may not be detected if the "caps lock" or "Num lock" modes are on. Be sure to release these modes first.

2 Output

The basic mechanism in `simplecpp` is to execute graphics commands as soon as they are issued. Thus if you create a few objects, move them, these are executed sequentially. However you often want the effect of commands to appear simultaneously on the screen, e.g. you might want all planets to move simultaneously as they rotate around the sun.

The command `beginFrame();` will cause all changes to the screen to be temporarily withheld. If you subsequently move an object or change its colour and so on, these will not be shown on the screen.

The command `endFrame();` resumes screen updates. Thus all changes made to the objects between a `beginFrame()` and the following `endFrame()` will appear to happen at once.

If you only have a few objects on screen, then it may not be necessary to use `beginFrame` and `endFrame`; the updates will appear rapidly and may seem simultaneous.