Mobile Information Device Programming (12)

Lecturer: Alireza Mousavi
School of Engineering & Design
www.brunel.ac.uk/~emstaam

Game Actions Detection

- Key codes provide event handling on **Canvas**

- An event is triggered via *keyPressed( ), keyReleased( )* and *keyRepeated( )* [passing key code]

Key codes and game actions

There are to ways to map key codes and game actions:

Firstly,

- At the initialisation stage we use *getKeyCode( )* to request and store key codes for each game action

- Define *keyPressed( ), keyReleased( )* and *keyRepeated( )* based on the key code

Key codes and game actions cont.

Example:

```
// at initialisation stage
Rightkey = getKeyCode(RIGHT);
Leftkey = getKeyCode(LEFT);
Downkey = getKeyCode(DOWN);
…
// runtime
Protected void keyPressed(int  keyCode){
    if (keyCode == Downkey)
    moveDown( );
    else if (keyCode == Leftkey)
    moveLeft( );
    …
    }
```

Key codes and game actions cont.

Secondly,

- Convert the incoming key code into game action inside the *keyPressed( ), keyReleased( )* and *keyRepeated( )*
- Branch based on the game action

Key codes and game actions cont.

Example

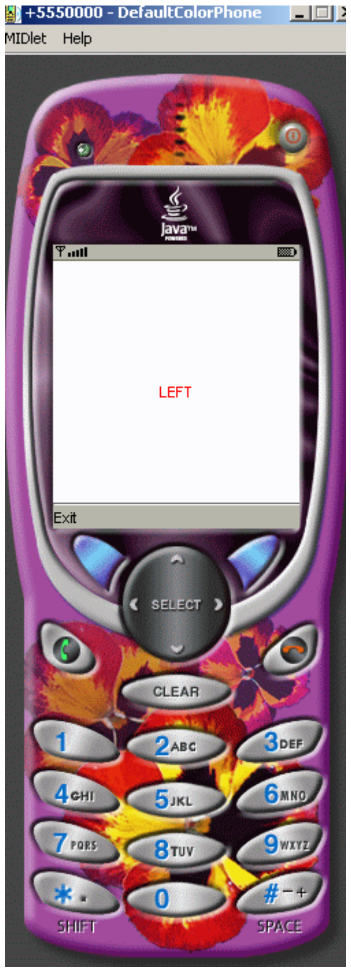
```
protected  void keyPressed(int  keyCode) {

Switch (getGameAction(keyCode))
{
      case Canvas.DOWN:
      moveDown( );
      break;
      case Canvas.LEFT
      moveLeft( );
      break;
      …
}
```

Game Actions Exercise (12-1)

- Write an application associating text to game actions
- Use *keyPressed( )* will catch events and convert the incoming key codes into a text string.
- Use the *paint( )* method to print the text on canvas

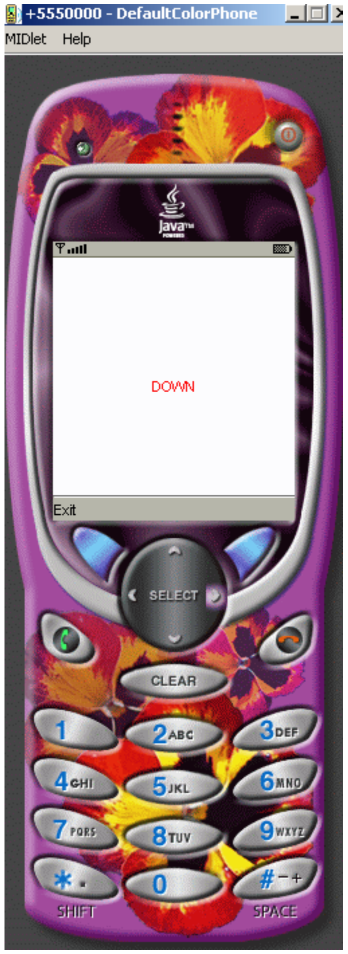
Result


+5550000 - DefaultColorPhone

MIDlet Help

LEFT

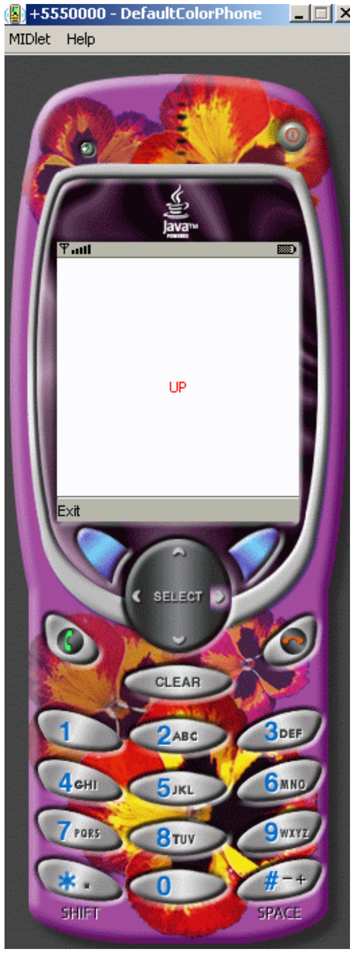
Exit


+5550000 - DefaultColorPhone

MIDlet Help

DOWN

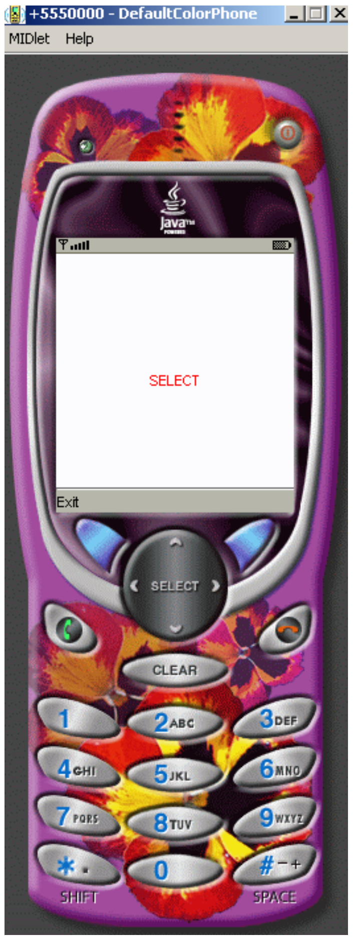
Exit


+5550000 - DefaultColorPhone

MIDlet Help

UP

Exit


+5550000 - DefaultColorPhone

MIDlet Help

SELECT

Exit

Interaction with other features

In cases a device may have features such as mouse or touch screen:

- These are called Pointer Events
- Similar to other key codes
- Methods used are: *pointerPressed( ), pointerReleased( )* and *pointerDragged( )*
- They are place holders and need to be overridden if the application supports pointer events.

Pointer event methods

Method	***Description***
boolean *hasPointerEvents( )*	Does the device support a pointer
boolean *hasPointerMotionPoint( )*	Does the platform support pointer motion
void *pointerDragged(int x, int y)*	Invoked when pointer dragged
void *pointerPressed(int x, int y)*	Invoked when pointer pressed
void pointerReleased(int x, int y)	Invoked when pointer released
Source: Core J2ME, J. W. Muchow, Sun Microsystems	

Example

```
protected  void pointerPressed(int x,  int y){

Startx = x;    // start coordinate
Starty = y;
}
protected  void pointerDragged(int x, int, y){

Currentx = x;  // current location of the pointer x axis
Currenty = y;  // current location of the pointer y axis
}
protected  void  pointerReleased(int x, int y){
Finalx = x;  // final position of the pointer x axis
Finaly = y;  // final position of the pointer y axis
 }
```

Assignment Ex.12-2

- Freehand movement of pointer on canvas and drawing
- Track the pointer as it moves along the canvas
- When the mouse button is down and dragged a line will be drawn.
- When the mouse is released the drawing is stopped

Explanations E12-2

- When the user clicks on the mouse button or the stylus for touch screen devices the *pointerPressed ( )* method is called

```
protected void  pointerPressed(int x, int y){
    startx = x;
    starty = y; }
```

- Each movement of the pointer goes to – the current x and y position are saved and the display is asked to repaint

```
pointer void  pointerDragged(int x, int y){
    currentx = x;
    currenty = y;
    repaint( ); }
```

- This repaint request will result into a call to *paint( ),* hence a line is drawn from the starting point to the current position – go to next page

Explanations E12-2 cont.

```
g.drawLine(startx, starty, currentx, currenty);
// new starting point
startx = currentx;
starty = currenty;
```

- The current x and y position will become the starting point for the next line. Hence in the *paint( )* startx and starty are equalled to the current position.
- To start a new free hand(doodle) select the Clear command
- If cmClear is selected then a check is made in *paint( )* if the result of:

  if(clearDisplay)

Is **true** the display is cleared by drawing a white rectangle the size the Canvas.

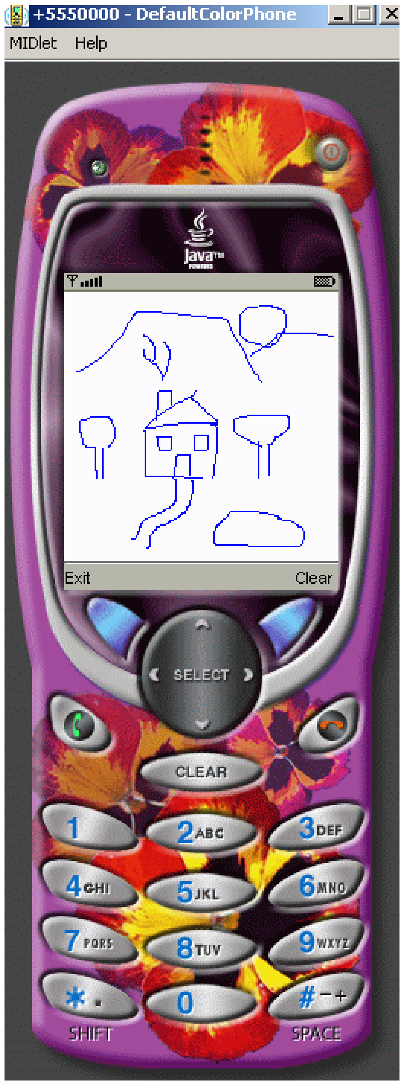
- Also the positions are reset – see next page

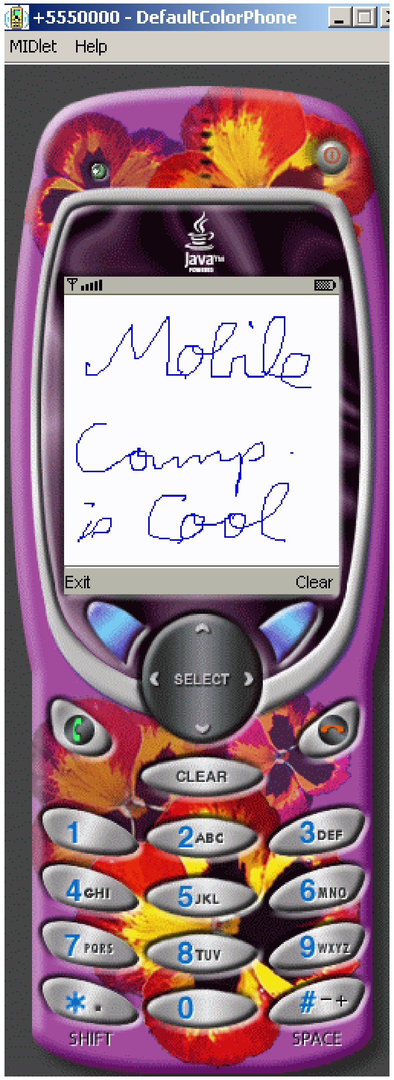
Explanations E12-2 cont.

```
// clear the background (draw a white recatngle)
      if (clearDisplay)
      {
        g.getColor(255,255,255);
        g.fillRect(0, 0, getWidth( ), getHeight( );

        clearDisplay = false;
        startx = starty = currentx = currenty = 0; // all to the 0, 0 coordinates
        return ( );
    }
```

Result


+5550000 - DefaultColorPhone
MIDlet Help
Exit
Clear
SELECT
CLEAR
SHIFT
SPACE


+5550000 - DefaultColorPhone
MIDlet Help
Mobile
Comp.
is Cool
Exit
Clear
SELECT
CLEAR
SHIFT
SPACE