Computer Programming using C
Lecture 5
(optional lecture)

Dr Julian Miller
Room P/M/101
Ext 2383; E-mail: jfm@ohm
http://www.elec.york.ac.uk/intsys/users/jfm7/
Drawing a projectile: preliminaries

- We want the user to enter a speed for the projectile
- The projectile leaves the stick man’s hand
- We want to create a series of horizontal positions and find what the corresponding vertical position of the projectile is.
- When thrown the projectile moves horizontally at a fixed speed equal to the horizontal component of its initial speed
- Gravity acts downwards so the projectile accelerates, that is to say its vertical speed component increases
Calculating the time to reach a horizontal position

time taken to get from initial_pos_x to pos_x is given by

\[
time = (pos_x - initial_pos_x) / vel_x
\]
Calculating the vertical position of the projectile after a certain time

Newtonian kinematics says that if a body is initially moving a speed, \( u \), and has acceleration, \( a \), then its position (assuming all in a straight line) after a time, \( t \), is given by

\[
y = ut + \frac{1}{2} at^2
\]

So using our variable names:

\[
\text{pos}_y = \text{initial}_\text{pos}_y - (\text{vel}_y \times \text{time}) + \text{GRAVITY} \times \text{time} \times \text{time}/2
\]
Incrementing horizontal position and drawing the projectile path

• To draw the projectile path we use little straight lines between positions along the path.
• We increment \texttt{pos\_x} and calculate the new \texttt{pos\_y} and use \texttt{lineto} to draw a straight line on the screen between these points
Simple projectile code fragment

```c
pos_x=initial_pos_x;
moveto(initial_pos_x,initial_pos_y);
do
{
    time = (pos_x - initial_pos_x) / vel_x;
    pos_y = (int)(initial_pos_y - (vel_y * time) + (gravity * time * time)/2);
    lineto(pos_x,pos_y);
    pos_x++;
}
while (pos_x < x_window);
```
The projectile shouldn’t go beneath the ground

- The ground is defined as any \((x, y_{\text{ground}})\) where \(x\) is any allowed \(x\) value and \(y_{\text{ground}}\) is a constant.

```c
pos_x=initial_pos_x;
moveto(initial_pos_x,initial_pos_y);
do
{
    time = (pos_x - initial_pos_x) / vel_x;
pos_y = (int)(initial_pos_y - (vel_y * time) + (GRAVITY * time * time)/2);
    if (pos_y <= y_{ground})
    {
        lineto(pos_x,pos_y);
    }
    else
        break;
}while (pos_x < X\_WINDOW);
```
Exercise 3.7: Making the stick man move (outline)

• To make the stick man appear to move we need to draw over the old position in black and then redraw the new position

```c
/* draw stick man */
/* print instructions to user */
/* get key_pressed */
while (key_pressed!=RETURN) // replot stick man unless RETURN pressed
{
    /* redraw stick man in black */
    /* move head if arrow keys pressed */
    /* redraw stick man in new position in RED */
    /* read keyboard again */
} /* end of while loop */
```
Exercise 3.7: The generic stick man

/* stick man constants */
define HEADSIZE 20
define ARMLength 20
define BODYSIZE 80
define ARMPOSITION 40
define STRIDE 30
define LEG_LENGTH 40
define STICK_MAN_HEIGHT HEADSIZE+BODYSIZE+LEG_LENGTH

circle(x_head, y_head, HEADSIZE);
/\ Draw body /
line(x_head, y_head+HEADSIZE, x_head, y_head+BODYSIZE);
/\ Draw arms /
line(x_head-ARMLength, y_head+ARMPOSITION, x_head+ARMLength, y_head+ARMPOSITION);
/\ draw legs /
line(x_head-STRIDE, y_head+BODYSIZE+LEG_LENGTH, x_head, y_head+BODYSIZE);
line(x_head+STRIDE, y_head+BODYSIZE+LEG_LENGTH, x_head, y_head+BODYSIZE);
Exercise 3.7: Moving stick man

```c
setcolor(RED);
/* draw stick man */
printf("Move the stick man by pressing:<n");
printf("< for left and -> for right and return to accept<n");
key_pressed=getch();
key_pressed=getch(); //arrows are extended keys so need two calls
while (key_pressed!=RETURN) // replot stick man unless RETURN pressed
{
    /* redraw stick man in black */
    setcolor(BLACK);
    /* draw stick man */

    /* move head if arrow keys pressed */
    if (key_pressed==RIGHTARROW)
        x_head=x_head+2;
    else if (key_pressed==LEFTARROW)
        x_head=x_head-2;
    else
        break; // break from while loop if non-arrow pressed

    /* redraw stick man in new position in RED */
    setcolor(RED);
    /* draw stick man */

    /* read keyboard again */
    key_pressed=getch();
    key_pressed=getch();
} /* end of while loop */
```
Exercise 3.8: Giving the user three goes

```c
printf("welcome to the stick man game\n\n");
for (go = 0; go < num_goes; go++) {
    /* tell user what go he/she is on */
    /* put rest of code here */
}
```
In the lab this week (lab 5)

• Functions
  – Are special collections of instructions that can be ‘called’ by programs
  – They make programming much easier
  – If a body of code needs to be repeated then it should be placed in a function
  – You have used some functions already (scanf, getch, printf)
  – In this lab you learn how to write your own