An Introduction to Programming though C++

Abhiram G. Ranade

Ch. 5: Simplecpp Graphics

Simplecpp Graphics

Much more is possible besides moving turtles.

- Other shapes besides turtles.
- Colour and size can be changed.
- Absolute coordinates (relative to canvas: drawing area) can be used
- Graphical input also possible

Starting up general graphics

- Use `initCanvas`, not `turtleSim`:
- First form

```
initCanvas();
```

  - Opens a window (“canvas”) for drawing.
- Second form:

```
initCanvas(name, w, h);
```

  - name: quoted string, will appear on window.
  - `w, h`: width, height of window

Co-ordinate system

- The canvas origin is in top left corner.
- x axis goes rightward.
- y axis goes downward.
- New commands are available which use the coordinates.

Creating multiple turtles

- General form

```
Turtle n1,n2,…,nk;
```

- Creates k turtles having names `n1,…,nk`. Any identifier can be used as a name.
- Initially all turtles are at the center of the canvas.
- You can selectively move turtle `ni` by writing

```
ni.forward(50);
```

- This causes turtle named `ni` to move forward 50 pixels.
- Other commands, `left`, `right` can also be used instead of `forward`.

Drawing 3 octagons using 3 turtles moving simultaneously

```
main_program{
    initCanvas();
    Turtle t1, t2, t3;
    t2.left(120); t3.left(240);
    repeat(8){
        t1.forward(100); t2.forward(100); t3.forward(100);
        t1.left(360/8); t2.left(360/8); t3.left(360/8);
    }
}
```

Demo

Creating graphical objects

- Basic form:

```
shape-type name(arguments);
```

- Creates graphical object of type `shape-type`. Possible types:
  - `Circle, Rectangle, Line, Text`
- `name` : Name given to created object.
- Created object can be manipulated by writing `name.forward(100);` etc.
- Multiple objects of same type can be created by giving more comma separated names with arguments.
- Pens of non turtle shapes are up by default.

Circles

- Supply 3 arguments: x, y coordinates of center, and radius.

```
Circle c1(100,100,10), c2(100,100,20);
```

- Creates circles of radius 10 and 20, both centered at (100,100). The circles are named c1, c2.

Rectangles

- Supply 4 arguments: x, y coordinates of center, and width and height.

```
Rectangle r1(200,100,20,40);
```

Lines

- Supply x, y coordinates of first endpoint, then second endpoint.

```
Line, l1(20,20,20,30), l2(25,15,25,25);
```

- Creates two lines, together forming a little “+” symbol.

Text

- Supply the x, y coordinates where the text is to be centered, and then the text itself in quotes.
- Commands `textWidth(t), textHeight(t)` evaluate to the width and height of the text `t` in pixels.
- Example:

```
Text t(100,100,”C++”);
Rectangle r(100,100, textWidth(“C++”)+4,
                     textHeight(“C++”)+4);
```

- Writes “C++” at coordinates (100,100) and puts a rectangle around it.

Commands allowed on shapes

- Let s be any shape created earlier.

```
s.moveTo(x,y);  // moves s to (x,y)
s.move(dx,dy);  // moves s by dx,dy
s.scale(factor);// scales s by factor
s.rotate(angle);// rotates s by angle
                // must be in radians
```

- rotation and scaling cannot happen on text.

Colors

When a graphics object it created, it has the colour black, except Turtles, which are red. But the color can be changed.

```
s.setColor(col);
```

- Changes the color of s to `col`. `col` must be specified as `COLOR(“red”)` and so on. Common color names are recognized.

```
s.setColor(red, green, blue);
```

- Here `red, green, blue` must be values between 0 and 255. The object s gets the color obtained by mixing the corresponding shades.

Filling Rectangles and Circles

```
s.setFill(v);
```

- Allowed only when `s` is a `Rectangle` or a `Circle`.
- If `v` is `true`, then the interior of `s` is filled with its color.
- If `v` is `false`, then the interior is left white.

Tracking a shape

- `s.getX()` returns the current x coordinate of s.
- Likewise `s.getY()`
- `s.getOrientation()` returns the current orientation, i.e. the angle through which s has been rotated so far.
- `s.getScale()` returns the current scale factor used for s.

Imprinting a shape

- `s.imprint();` causes an image of s to be permanently drawn on the canvas at its current position, i.e., even after s moves, the image will be present.
- If you wish to create a static picture involving 100 circles:
  - Create one circle
  - Move it to appropriate positions
  - Imprint at each position.

Graphical input

```
getClick()
```

- causes the program to wait until the user clicks on the screen
- Then it returns the value 65536x + y, where x, y are the coordinates of the cursor at the position of the click.
- Note that we can get back the coordinates by writing

```
int w = getClick();
int x = w/65536, y = w % 65536;
```

- This works because the coordinates are at most a few thousand, much smaller than 65536.
- You may just write

```
getClick();
```

- In this case, the program just waits for the user to click.

Example

```
main_program{
   initCanvas(“Projectile”, 500, 500);
   int start = getClick();
   Circle p(start/65536, start % 65536, 5);  // at click position
   p.penDown();                                // let us see its path
   double vx=1, vy=-1, gravity=0.01;
   repeat(500){
      p.move(vx, vy);
      vy += gravity;
      wait(0.01);
   }
   getClick();    // wait for the user to click.  Only then terminate.
}
// Will show a circle move as if thrown against gravity
// from the click position.
```

Demo

Exercises

- Draw a plot of y = sin(x). Use imprinting to draw many small lines.
- Suppose a man is walking from the origin in the positive x direction. In a single time unit the man walks 3 pixels. A dog is at point (0,300). At the beginning of each time unit, the dog turns towards the man and in the time unit walks 6 pixels. Show what happens in 50 time units.
- Create nice designs.
- Show the movement of a projectile having horizontal velocity v cos(t) and vertical velocity v sin(t) for fixed v and different t. You should be able to observe that when t is 45 degrees, the project goes the longest distance.

Summary

- Graphical shapes with names can be created.
- Some basic shapes such as circles, lines, and rectangles are only allowed. These can be moved around/rotated/scaled on the screen.
- Text is also allowed, and may even be moved around on the screen.
- The book gives more details.
- Polygons can also be drawn, as will be seen in a later chapter.
- More sophisticated graphical input will also be seen in a later chapter.
- In the later chapters you will be able to build on this to create graphical editors, sophisticated animations.