simplecpp Graphics

Abhiram Ranade

Outline

- Many turtles
- Other shapes
- Operations on shapes
- Projectile motion
- Best fit line

initCanvas

initCanvas(“window name”,w,h);

- Use instead of turtleSim();
- Opens a window of given name, width and height.
- Turtle not created automatically.
- But you can create turtles and other shapes as you wish.
- closeCanvas() to remove canvas.

Coordinate system

- Origin is at top left corner
- x axis goes to right
- y axis goes goes downward.

Multiple turtles

Turtle t1, t2, t3;

- Creates 3 turtles, called t1, t2 and t3.
- All are initially at the center.
- To command a turtle t, use t.command, e.g.
  - t1.forward(100);
  - t2.right(45);

Other Shapes

- General form:

```
Shape name-of-object(arguments);
```

- Example: circle

```
Circle c1(cx, cy, radius);
```

  - c1 : name of circle
  - cx,cy : coordinates of center (double)
  - radius: radius of circle (double)
- Creates that shape on the screen.

Executing commands on Shapes

shape.command(arguments)

- Example: c1.forward(100)
  - every object is created pointing in the positive x direction.

Other Shapes

- Rectangle r2(cx,cy,w,h);
  - r2 : name of rectangle (Axis parallel, cannot be rotated)
  - cx,cy : as above
  - w,h : width and height
- Line l3(x1,y1,x2,y2);
  - coordinates of endpoints.
- Text t4(x,y,”message”);
  - “message” appears centered at x,y

Commands allowed on shapes

- moveTo(x,y) : center point of object moves to absolute coordinates (x,y)
- move(dx,dy) : object moves by given increment in x, y directions.

```
c1.move(3,5);  t4.moveTo(300,400);
```

- In both cases line is drawn if pen is down.

Commands allowed on shapes

- scale(double relative-factor)

```
c1.scale(2);        // doubles radius
```

- setScale(double absolute-factor)

```
Circle c1(100,100,5);
c1.scale(2);        // radius 10
c1.scale(3);        // radius 30
c1.setScale(1.5); // radius 7.5
```

Commands allowed on shapes

- imprint() : print on the canvas. Will remain even after the shape moves.
- setColor(COLOR(“name-of-colour”)
- setColor(COLOR(redV,greenV,blueV)) ;

```
c1.setColor(COLOR(“blue”));
c1.setColor(COLOR(255,255,0); //yellow
```

- setFill() : interior of object will be filled with color of object. Otherwise only border has that colour.

Resetting a shape

```
Rectangle r1(100, 200, 20, 20);
wait(5);
r1.reset(100, 200, 10, 40);
```

- reset: same parameters as at creation. Recreates the object.
- In this case square will appear to flatten.

Graphical input

```
int clickval;
clickval = getClick();
```

Wait until user clicks on simplecpp window.

click-val will equal

x-coordinate of click * 65536

+ y-coordinate of click.

Input Example

```
main_program{
  initCanvas();
  int cval = getClick();
  Circle c(cval / 65536, cval % 65536,
              10);
  // circle of radius 10 at click position.
  wait(5);
}
```

Projectile motion

```
main_program{
  initCanvas(“Projectile”, 500, 500);
  int cval = getClick();
  Circle projectile(cval/65536, cval % 65536, 5);
  double vx = 1, vy = -5; // up
  repeat(100){
    projectile.move(vx, vy); wait(0.1);
    vy += 0.1; // gravitation;
  }
}
```

“Best fit” line

Input: points in the plane.

(x1,y1), (x2,y2), ...

Output: m, c, where $y=mx+c$ is the equation of the “best” line representing the points.

“line should be as close to all points as possible”

Algorithm Outline

Point: (xi, yi)

Line: y = mx + c

Error of point: (yi – m xi - c)^2

Total Error = sum of per point error.

Choose m, c such that total error is minimized.