Scratching Sprite.java

Functions and Properties reference

We make new Sprites by *declaring* them at the top of our code, just like a *variable.*
Example:

Sprite mySprite = new Sprite(this);   // create a new Sprite

Properties are accessed a command such as

myVariable = SpriteName.property;

int Sprite.rotationStyle;

This defines how the Sprite turns, just like the Scratch buttons. This can be any of
rotationStyle_AllAround
rotationStyle_LeftRight
rotationStyle_DontRotate
Example call:
mySprite.rotationStyle=mySprite.rotationStyle_LeftRight;

int Sprite.costumeNumber;

This holds the number of the current costume.

int Sprite.numberOfCostumes;

This holds the number of costumes the Sprite has

int Sprite.ghostEffect;

The Ghost Effect ranges from 0 (invisible) to 255 (solid).
Example:
mySprite.ghostEffect = 125; // approximately half-visible

float Sprite.size;

Sprite.size dictates the size of the Sprite by percentage. The default value of 100 is the full size of the image file. 50 is half size. 200 is double. And so on.

boolean Sprite.visible;

You can hide a sprite by setting *mySprite.visible=false;* show by *mySprite.visible=true;*

PVector Sprite.pos;

A “PVector” object contains positioning information for the Sprite. Rather than modifying the vector information directly, mostly you will want to read and set its *x* and *y* properties, as below:
int mySprite.pos.x
int mySprite.pos.y

float direction;

Direction is in degrees. Unlike Scratch, *0 degrees points to the right in Processing.*
So while the use of direction is familiar to you, it will also be a little different.

Functions and Procedures

A *void* function is called simply, such as

mySprite.move(10);   // move the Sprite 10 pixels in the current direction*

Others, such as *int* functions, return *calculated values*, much like a *variable*.

int myVariable = mySprite.distanceTo(otherSprite);
// store the distance between mySprite and otherSprite

void Sprite.update();

Takes no arguments. This performs the actual rendering of the Sprite.
It is necessary to call Sprite.update() once per draw() loop, *after Stage.update();* in order to make a Sprite appear on the Stage.

Example call:
mySprite.update(); // display Sprite on stage during the current loop

void move(int distance);

Move the Sprite *distance* pixels in the current direction.

Example call:
mySprite.move(10); // move sprite 10 pixels in current direction

void goToXY(int x, y);

Move immediate to the grid co-ordinates specified.

Example call:
mySprite.goToXY(0,0); // return to center of Stage

void goToSprite(Sprite target);

Move immediately to the position of the target sprite.

Example call:
mySprite.goToSprite(otherSprite); // move to the position of the Sprite otherSprite

void loadDefaultCostumes();

This is called automatically by the initializer, and loads the Scratch default cat costumes.
You can modify this to replace the images with your own Sprite Art.

Example call:
You should never have to call this function.
loadDefaultCostumes() is run by the initializer when you declare a new Sprite object.

void addCostume(string filepath);

This adds a costume to the Sprite from an image in your Sketch folder. Your Sprites may get complicated with a lot of frames, so be sure to name and sort your images into folders! For example, in a game where a hungry Spider hunts for her dinner, you might have the images,
Documents/Processing/mySketch/art/player/spider_walking_1.png
Documents/Processing/mySketch/art/player/spider_walking_2.png
Documents/Processing/mySketch/art/player/spider_jumping_1.png
Documents/Processing/mySketch/art/player/spider_jumping_2.png
Documents/Processing/mySketch/art/enemies/ant_walking_1.png
Documents/Processing/mySketch/art/enemies/ant_walking_2.png
Documents/Processing/mySketch/art/enemies/bee_flying_1.png
Documents/Processing/mySketch/art/enemies/bee_flying_2.png

After you draw (or download and edit) and place the files, add them to your Sprite with:
addCostume("Documents/Processing/mySketch/art/player/spider_walking_1.png");
addCostume("Documents/Processing/mySketch/art/player/spider_walking_2.png");
Et cetera

void nextCostume();
void previousCostume();

Changes to the next or previous costume.
If you are on the first or last costume, these will intelligently loop to the opposite.

Example call:
mySprite.nextCostume();
mySprite.previousCostume();

void setCostume(int newCostumeNumber);

Change to a specific costume. This means remembering costume numbers.
Or you can add *constant values* which give those numbers easy names to remember.
Define them as follows
public *static* int playerWalkingCostume=0;   // *static* means the value will not change
public *static* int playerJumpingCostume=2;   // *static* means the value will not change

Example call:
mySprite.setCostume(3);
mySprite.setCostume(mySprite.playerJumpingCostume);

void show();
void hide();

These functions Show and Hide a Sprite, like the Scratch blocks. These are provided to make adapting existing projects more straightforward. You could just as easily access the boolean *Sprite.visible* to set visibility.

Example call:
mySprite.show();   // is the same as mySprite.visible = true;
mySprite.hide();   // is the same as mySprite.visible = false;

void turn(int angle);
void turnLeft(int angle);
void turnRight(int angle);

These turn the Sprite by degrees, as in Scratch.

Example calls:

```
mySprite.turn(-45);          // turning 45 degrees left
mySprite.turn(45);           // turning 45 degrees right
mySprite.turnLeft(45);       // turning 45 degrees left
mySprite.turnRight(45);      // turning 45 degress right
```

void pointTowardsXY(int x, y); // point towards an arbitrary grid position
void pointTowardsSprite(Sprite target); // point Sprite towards a target Sprite
void pointTowardsMouse(); // point towards the mouse cursor
void pointInDirection(int angle); // point in a specific direction.

Set the Sprite's direction.

Example calls:

```
mySprite.pointTowardsXY(0,0);                 // point towards center of Stage
mySprite.pointTowardsSprite(otherSprite);     // point towards Sprite named otherSprite
mySprite.pointTowardsMouse();                 // point towards the mouse cursor
mySprite.pointInDirection(0);                 // point right
mySprite.pointInDirection(90);                // point up
mySprite.pointInDirection(180);               // point left
mySprite.pointInDirection(270);               // point down
```

float distanceToXY(int x,y); // returns distance to arbitrary position
float distancetoSprite(Sprite target); // returns distance to another Sprite
float distanceToMouse(); // returns distance to mouse cursor

These functions return distances to objects.

Example call:

```
float distanceToCenter = mySprite.distanceToXY(0,0);               // store distance to center
float distanceToSprite = mySprite.distanceToSprite(otherSprite);   // store distance to sprite
float distanceToMouse = mySprite.distanceToMouse();                // store distance to mouse
```

boolean touchingSprite(Sprite target);

This function returns a **true** value when the two Sprites are touching.
This performs a simple rectangular hit box check, which gives good but imperfect results.

Example call:

```
bool spritesAreTouching = mySprite.touchingSprite(otherSprite); // store "touching" value
if (spritesAreTouching) mySprite.hide();              // hides mySprite if touching otherSprite
else mySprite.show();                                 // shows mySprite if not touching
```