This is still fairly accurate. Basically, GameActors are sprites, and that's what you need to know. *Much* more to come later¼

GameActor

INHERITS FROM	Object
DECLARED IN	GameActor.h

CLASS DESCRIPTION

A GameActor is an abstract class designed to handle a moving object for an animation program. A particular subclass would then have certain methods overridden so as to allow it to move about and render itself in it's own unique way. However, since all subclasses inherit the basic movement and rendering methods, it doesn't matter to which type of GameActor you send a message...to the caller, they all appear to just be actors.

The framework provided requires the subclass to implement the **-move:** and **-renderAt::move:** methods in order to create a GameActor that actually does something. The **-move:** method sets up instance variables *px* and *py* so that the GameActor knows which way to move for the next frame. The **-renderAt::move:** method renders the GameActor inside some subclass of View. You must **-lockFocus** on the View subclass before attempting to render the GameActor.

To use the GameActor subclass, you instantiate it and then send it messages when you want it to move or render itself. Typically, you will call **-move:** to set up movement for all the GameActors and then call **-renderAt::move:** to draw them all inside of the main game screen. Note that **-move:** doesn't actually cause the GameActor's coordinates to change±it only decides how they will change. This way, all GameActors that decide how to move based upon locations of other GameActors won't cause wierd bugs due to some GameActors moving before other GameActors. To make the GameActor actually move, you

must call **-moveOneFrame**. (Note that **-renderAt::move:** will do this if you specify a value of YES for *moveOk*.)

At times the controller or another GameActor will need to know where the GameActor is located or was last drawn. These coordinates are available via the **-xpos**, **-ypos**, **-lastAt::**, and **-at::** methods.

INSTANCE VARIABLES

Inherited from Object — Class isa;

Declared in GameActor

```
int   myX, myY;
int   lastx, lasty;
int   px, py;
id    gameView;
```

myX, myY — Current location of the GameActor.

lastx, lasty — Where the GameActor was last rendered.

px, py — Deltas specifying how the GameActor will move in the next frame.

gameView — A GameView subclass. Not actually used by this abstract object, but it is provided because subclasses will often need a connection to a parent GameView subclass for some reason or another.

METHOD TYPES

Initialization
- init
- gameView
- setGameView:

Moving and Animation
- move:
- moveOneFrame

Displaying
- renderAt::move:

Current Location
- at::
- lastAt::
- xpos

- ypos

INSTANCE METHODS

at::

- **at:**(float *)*xx* **:**(float *)*yy*

Returns by reference the current location of the GameActor. Returns **self**.

See also: **± lastAt::**, **±xpos**, and **±ypos**

init

- **init**

Designated initializer for the GameActor. Clear all the coordinates and deltas. Returns **self**.

See also: **± setGameView:**and **±gameView**

gameView

- **gameView**

Returns the current value of the instance variable **gameView**.

See also: **± setGameView:**

lastAt::

- **lastAt:**(float *)*xx* **:**(float *)*yy*

Returns by reference the last location where the GameActor was rendered. Returns **self**.

See also: **± at::**, **±xpos**, and **±ypos**

move:

- **move:***sender*

Sets the deltas (**px**, **py**) which tell the GameActor where to move for the next frame of animation.

Subclasses of GameActor are expected to implement this method; in the generic GameActor, it does nothing. Returns **self**.

See also: **± moveOneFrame**, and **±renderAt::move:**

moveOneFrame

- **moveOneFrame**

Actually causes the GameActor to change coordinates. Usually called by **±renderAt::move:**. If overridden, the subclass must call the **super** method at some point. Returns **self**.

See also: **± moveOneFrame**, and **±renderAt::move:**

renderAt::move:

- **renderAt:**(int)*posx* **:**(int)*posy* **move:**(BOOL)*moveOk*

Renders the GameActor. This method assume that you have first locked focus on some View or subclass of View. Subclasses of GameActor are expected to implement this method. (A subclass should call the **super** method first, before implementing subclass-specific code.)

The coordinates *posx* and *posy* are offsets into the View subclass which currently is focused; this allows the actual game screen to have borders, etc.; the GameActor assumes that the coordinates (0,0) are the lower left corner of the view; setting *posx* and *posy* non±zero allows you to change this.

The *moveOk* flag tells the GameActor whether or not it needs to move itself before rendering. See the description above for why movement is delayed until rendering. Note that if *moveOk* is set to NO, then you can re-render a game screen without moving the GameActor.

Returns **self**.

See also: **± move:** and **±moveOneFrame**

setGameView:

- **setGameView:***newGameView*

Sets the instance variable **gameView** to *newGameView.* Returns **self**.

See also: **± gameView**

xpos

- (int)**xpos**

Returns the current x-coordinate of the GameActor.

See also: **± at::**, **±lastAt::**, and **±ypos**

ypos

- (int)**ypos**

Returns the current y-coordinate of the GameActor.

See also: **± at::**, **±lastAt::**, and **±xpos**