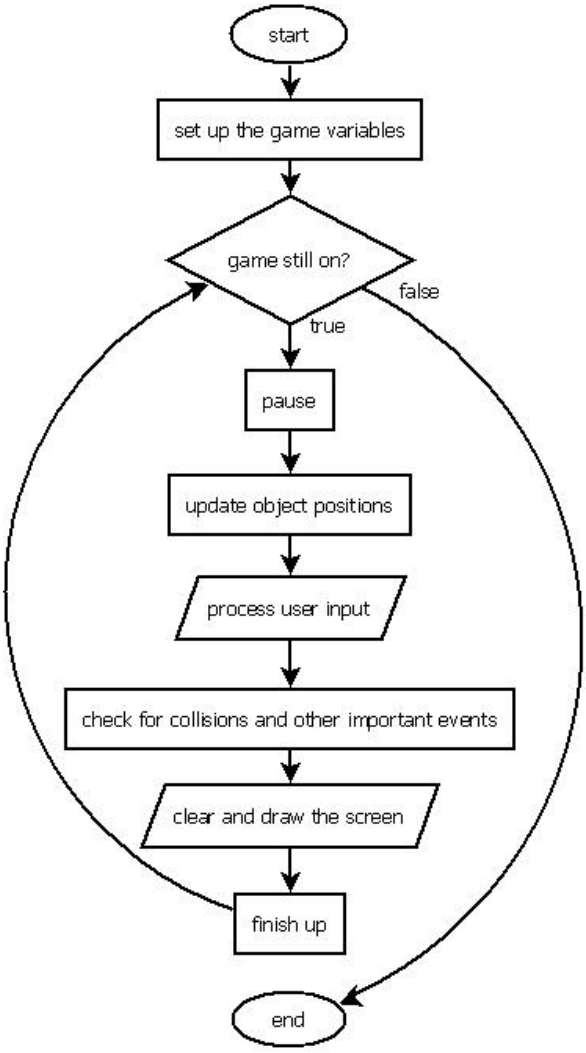
The GameEngine Design Pattern

So you want to program a live action game? You need two things - some better input methods, and a new design pattern. We'll start with the design pattern first…

1. The GameEngine Design Pattern

A game engine is a big loop, a bit like an animation loop. For smooth movement, you should loop about 30 times per second. Within the loop, you pause, update everything, respond to input, check for collisions and other special events, then redraw everything and loop back. The exact details will change from game to game, but the same approach will work for many games.



Notes

See **CrappyGame.java** for an example of this design pattern in action. The basic flow is simple, but getting the details right can be very tricky!

The part that says "check for collisions and other important events" might need some grade 10 math, namely Analytic Geometry.

For example, if you want to know whether two circles have intersected, you can compute the distance between them using the length of line segment formula. Then if the distance is small enough, they have collided.

When you clear and draw the screen, don't forget **synchronized**, or you will get lots of flicker.

2. Some Better Input Methods

Suppose you have a Pac Man controlled by the user and a monster chasing the Pac Man. You want the monster to keep moving even when the Pac Man is stopped. But there's a problem. When you get to the input step of the GameEngine loop, you only have input methods that halt the program and wait, like *c.getChar()*. We need some new methods that will not stop the program but just tell us what the user is doing *right now*.

Luckily for you, your favourite computer science teacher wrote those methods for the Console class a couple of years ago. **Remember that all these methods require a "c . " (or the console name, if it's not "c") in front of them.**

getKeyChar ()

> Returns a char representation of whatever key the user is holding down right now. Returns a null char (ASCII code 0) if nothing is held down.

getKeyCode ()

> Returns an ASCII representation of whatever key the user is holding down right now. Returns 0 if nothing is held down. This is useful for the arrow keys.

getLastKeyChar()

> Returns a char representation of the last key the user pressed. Returns a null char (ASCII code 0) if the user has not pressed anything since the program started.

getLastKeyCode()

> Returns an ASCII representation of the last key the user pressed. Returns 0 if the user has not pressed anything since the program started.

isKeyDown(char key)

> Returns TRUE if the key corresponding to the char in the "key" variable is currently held down. Otherwise, returns FALSE.

isKeyDown(int key)

> Returns TRUE if the key corresponding to the ascii code in the "key" variable is currently held down. Otherwise, returns FALSE.

See the example code on line (**CoolInputExamples.zip**) for examples of how to use these methods effectively.

3a. Special Keys

Not all keys have chars associated with them (like the arrow keys). In fact, not all keys have an ASCII code (like Shift or Ctrl). Luckily there are a bunch of constants defined in the system for this. Here are a couple of examples of how to use them…

```
int key = c.getLastKeyCode();

if (key == Console.VK_SHIFT)

      … blah blah blah…
```

```
    if (c.isKeyDown(Console.VK_LEFT))

         … blah blah blah …
```

In the examples above, Console.VK_SHIFT is they code for the Shift key and Console.VK_LEFT is the code for the left arrow key. For completeness, all the special constants are listed below.

3b. Special Key Constants

```
Console.VK_ALT
Console.VK_BACK_SPACE
Console.VK_CAPS_LOCK
Console.VK_CONTROL
Console.VK_DELETE
Console.VK_DOWN
Console.VK_END
Console.VK_ENTER
Console.VK_ESCAPE
Console.VK_F1
Console.VK_F2
Console.VK_F3
Console.VK_F4
Console.VK_F5
Console.VK_F6
Console.VK_F7
Console.VK_F8
Console.VK_F9
Console.VK_F10
Console.VK_F11
Console.VK_F12
Console.VK_HOME
```

```
Console.VK_LEFT
Console.VK_NUM_LOCK
Console.VK_NUMPAD0
Console.VK_NUMPAD1
Console.VK_NUMPAD2
Console.VK_NUMPAD3
Console.VK_NUMPAD4
Console.VK_NUMPAD5
Console.VK_NUMPAD6
Console.VK_NUMPAD7
Console.VK_NUMPAD8
Console.VK_NUMPAD9
Console.VK_PAGE_DOWN
Console.VK_PAGE_UP
Console.VK_PAUSE
Console.VK_PRINTSCREEN
Console.VK_RIGHT
Console.VK_SHIFT
Console.VK_TAB
Console.VK_UNDEFINED
Console.VK_UP
```