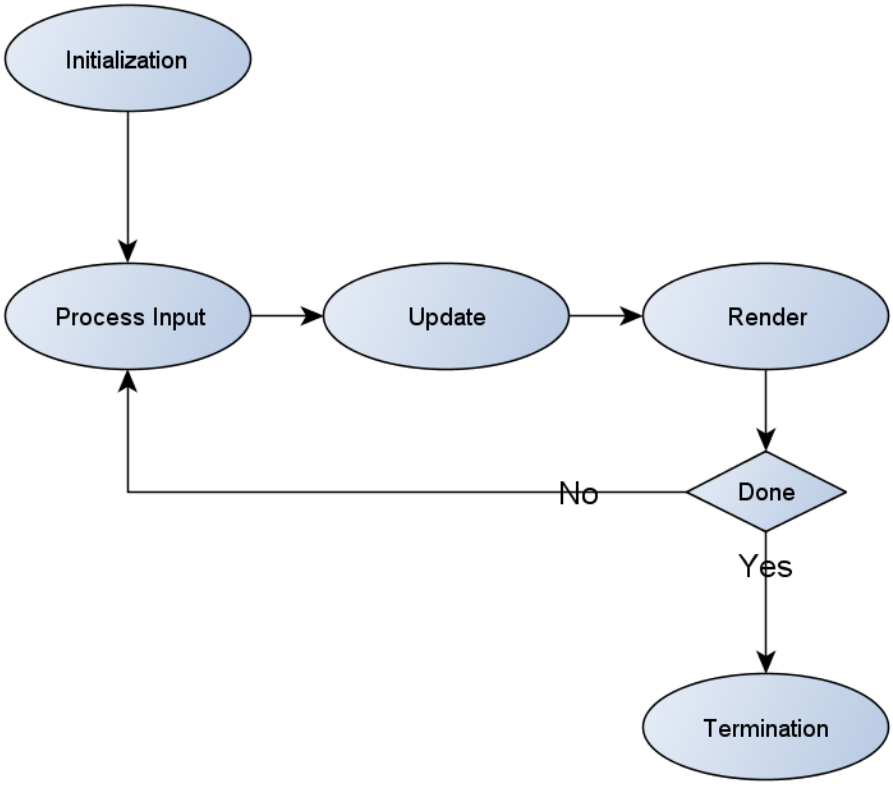
Game Architecture – The Game Loop

The Game Loop

With games we aren't (necessarily) worried about hogging up the CPU, we want the game to respond as quickly as possible. Therefore, the code for a game is always running in a continuous polling-like loop.

The game loop is what controls how the game is managed. A general overview of the standard game flow and loop is as follows...



The game loop in more detail...

Step 1: Initialize Game (Graphics, Files, Memory, Objects, etc.)

** Take Time Stamp ** (previous time)

Step 2: Collect Input

Step 3: Game Logic (update is based on delta time)

** Take Time Stamp ** (current time)

Artificial Intelligence
Collision Detection (note physics simulation)
Physics (might do multiple updates per frame)
Update Objects (move ships, players, animations)

Step 4: Render Game State

Step X: If fixed frame rate

Use a spin lock: wait until frame time expires

Step 5: Move current time stamp to previous time

Step 6: Check to see if done, if done move to Step 7, if not return to Step 2

Step 7: Termination (clean up after ourselves)

This form of a game loop will run as fast as possible. Therefore, the code must be aware of the timing between frames so that it knows how much to update objects between renders. This will result in a variable frame rate (this is okay and a generally a good thing, for most games).

- During initialization you grab a starting time (prev time)
- In the Game Logic section you then grab the current time
- The difference between current and prev time is how much to update the game objects

Sample JavaScript Code

```
function processInput(elapsedTime) {
  keyboard.update(elapsedTime);
}
```

```
function update(elapsedTime) {
  gameModel.update(elapsedTime);
}
```

```
function render() {
  gameModel.render(elapsedTime);
}
```

```
function gameLoop(elapsedTime) {
  processInput(elapsedTime);
  update(elapsedTime);
  render();

  requestAnimationFrame(gameLoop);
}
```

Timing

Timing plays an important role in games. We need to maintain an acceptable or fixed frame rate, move objects at some constant speed regardless of the frame rate, get things simulated and rendered and so on.

Frame Rate : Measured in Hz (frames per second - fps)

Frame Time/Delta Time : Amount of time between frames Δt. Average Delta Time is 1/fps

- The entire game simulation and rendering must take place in this amount of time
- If 30 fps, then each frame has 33.33 milliseconds for everything!

Moving Objects

For Example: Want to move an object at 20 (virtual) meters per second

- Bad Idea : Move at some number of (virtual) meters each frame
- Not as Bad Idea : Running Average : $x_2 = x_1 + v\,\Delta\, t_{ave}$ (where v is the velocity)
- Best Idea : Elapsed Time : $x_2 = x_1 + v\,\Delta\, t_{frame}$ (where v is the velocity)
- Fine Idea : Fixed Frame Rate : $x_2 = x_1 + v\frac{1}{fps}$