Peint

High-Performance,
Evented Canvas

Who am I?

I'm a developer for Blackcomb Games.

I build web & mobile games with HTML5.

Slotorama

Peint is...

- A Javascript graphics engine
- Supports HTML5 Canvas
- Component-based
- Good for any game (isometric, etc)

Philosophy

- Positioning and rendering should be handled separately
- High-frequency code should be as lightweight as possible
- Rendering should only happen when there are changes to render

Features

- Event-based rendering
- Object-based positioning
- Animation management
- Modular design will do CSS3 & WebGL

Event-based rendering

```
// Ugly nesting needed to ensure image
// is loaded before trying to draw it.
var img = new Image
img.onload = function () {
  function render () {
    requestAnimationFrame(render)
    // This image draws every frame,
    // even though it's state hasn't changed.
    context.drawImage(img, 0, 0)
  }
  render()
}
img.src = '/img.png'
```

Let’s clean that up

```
// Start canvas
var Game = Peint.all()
var canvas = new Game.Canvas('#viewport')
canvas.start()

// Load image and add to canvas
var img = new Game.Image({ url: '/img.png' })
canvas.add(img)

// Reposition later, redraw automatically
setTimeout(function () {
  img.set({ left: 200, top: 200 })
}, 5000)
```

Abstract rendering

- Support multiple render methods
- Add objects before loaded
- Change layer sorting
- Automate redraw

Animations

```
// Create animation
var animation = new Peint.Animation({
  url: '/img.png'
  , top: 330
  , left: 100
  , animation: {
    // Each column is a frame, and each
    // is a separate animation
    rows: 8, cols: 9
    // time between frames is adjustable
    // even while animating
    , duration: 40
  }
})
```

```
// play is called at image:loaded, let's pause
animation.on('image:loaded', function () {
  this.pause()
  // Let's unpause it later
  setTimeout(function () {
    animation.play()
  }, 1000)
})
```

Modularity

```
// Load selectively
var Canvas = Peint.require('canvas')
var Image = Peint.require('image')

// Start canvas
var canvas = new Canvas('#viewport')
canvas.start()

// Load and add image
var image = new Image({ url: '/img.png' })
canvas.add(image)
```

Advantages

- Easier to support multiple back-ends
- Execution order becomes irrelevant
- Only executes what it needs to

Alternatives

- Easel - monolithic, pollutes global
- Fabric - intended for drawing apps
- Paper - vector transform scripting
- Raphael - SVG, graphs and vector
- Processing - vector visualization

Limitations

- Primitives other than rectangle not yet implemented, low priority
- Currently canvas-only, needs at least CSS3 to better support mobile
- Alpha-stage, likely to change a lot and needs some better optimization

Planned

- Scaling & rotations (Partially done)
- Primitives, lines, etc
- CSS3 & WebGL renderers
- Better examples & tutorials

github.com/qard/peint