HTML5 Games with CreateJS

TWITTER: @DAVID_KELLEHER
DAVIDK.NET/GAME
CreateJS Features
HTML5 Canvas 2D Context Framework

Pros
- Bitmaps
- Vectors
- Animation
- Sprite Sheets
- Containers
- Flash Style Movieclips
- Inputs/Events
- Text
- Sounds
- Preloader

Cons
- No Level Editor
- No Physics
- No Particles
- No Virtual Camera
- Beta WebGL 2D Support
- No 3D Support
- No UI Controls
- No Networking
CreateJS Uses

Pros

- Leverage Adobe Flash skills and current support for Adobe Animate
- Create simple Flash-style games, banner ads, data viz, generative art
- Rapidly prototype 2D games
- Integrate with Angular, React, D3

Cons

- Not a high performance 2D library
- Not a true game development framework
- Not production ready - current version 0.8.2
CreateJS Performance

**Pros**

- Canvas hardware acceleration
- Caching of drawn objects - transform regions with complex vector graphics or many display objects

**Cons**

- No redraw regions - calculation time too expensive in JavaScript for arbitrary content
- Slow, very basic hit test
Architecture

- CreateJS - extends the HTML5 `<canvas>`
  - EaselJS
  - TweenJS
  - SoundJS
  - PreloadJS
Game Theory
Game Loop

Update Game State

UI: Game Output

Player: Game Input

Outcome: Win, Loss
Game Mechanics

- Strategy
- Dexterity
- Luck
Game Input and Events

User Input
- Mouse Move
- Click, Mouseup, Mousedown
- Drag and Drop, Swipe
- Key Input

Game Event
- Ticker
- Collision (Hit Test)
- Custom
Update Game State

Game Objects

• Add/Remove Objects
• Change Object Properties
• Tween Animations
• Update Score
• Test Win/Loss Condition
Dodgeball Inputs & Events

- Player's avatar tracks horizontal movement of mouse
- Balls can be clicked allowing players to catch them
- Game events: Detect collisions each frame between player and balls
Update Game State

- Add balls and remove when off court
- Move balls (motion tween)
- Change ball state when it is clicked, caught, or hits player
- Add point when ball is caught
- Change player state and end game when player hit by ball
Coding Strategy

Avoid creating a massively complex controller with all procedural coding in the game loop.
Object Oriented

- Blinky speeds up as you eat dots
- Pinky tends to move counterclockwise
- Inky makes unpredictable turns
- Clyde doesn’t always chase Pac-Man
Canvas Element
// HTML5 CANVAS ELEMENT
var canvas = document.createElement("canvas");
canvas.width = 600;
canvas.height = 400;

// --- Add canvas element to DOM
document.body.appendChild(canvas);
CreateJS Stage
CreateJS Stage

```javascript
// CreateJS STAGE // Uses 2D Canvas Context
var stage = new createjs.Stage(canvas);
stage.mouseMoveOutside = true;
```
CreateJS Ticker
EaselJS – Ticker Class

- Has a static interface
- Properties
  - framerate: frames per second
  - interval: milliseconds between ticks
  - maxDelta: maximum time between ticks
  - paused: pause the ticker
- Methods
  - reset(): stop ticker
  - init(): restart ticker
  - getMeasuredFPS(): returns actual ticks per second
  - getTicks(): total number of ticks broadcast
  - getMeasuredTickTime(): average tick execution time in ms
  - getTime(): milliseconds since ticker started
CreateJS Ticker

// CreateJS TICKER // Attaches Game Loop to Stage
createjsTicker.on("tick", stage);
createjsTicker framerate = 60;
Gym Background Object
Pass a URI to an image file, or a reference to an HTML Image, HTML Video, or HTML Canvas element to the constructor.

Don’t cache a single bitmap, unless you want to use a blur, alpha, or color filter, then you have to.

Properties:
- `image`: swap a new image into the object
- `mask`: shape image defining a clipping path
- `mouseEnabled`: enable/disable mouse event

Transform: x, y, scaleY, ScaleY, rotate, regX, regY
Gym Object

// GYM BACKGROUND OBJECT
var gym = new createjs.Bitmap("image/gym.jpg");

// --- Add gym to stage
stage.addChild(gym);
Score Object
EaselJS – Text Class

- Pass the text, optional CSS font, and optional CSS color to the constructor.

- Properties:
  - `text`: change the text
  - `color`: change the color
  - `font`: change the font
  - `outline`: add stroke to the text
  - `shadow`: add drop shadow to the text
```javascript
// SCORE OBJECT
var score = new createjs.Text(0, 'bold 50px sans-serif', '#FFF');
score.x = 20;
score.y = 20;

// --- custom properties
score.value = 0;

// --- Add player to stage
stage.addChild(score);
```
Player Object
// PLAYER GRAPHICS
var playerAlive = new Image();
playerAlive.src = "image/player.png";

var playerHit = new Image();
playerHit.src = "image/player-dead.png";
// PLAYER OBJECT
var player = new createjs.Bitmap();
player.image = playerAlive;
player.x = 232;
player.y = 275;
CreateJS Listeners

- `addEventListener`
  - pass event string and handler function
  - `addEventListener("click", handleClick);`

- `removeEventListener`
  - pass exact function used when added
  - `removeEventListener("click", handleClick);`
CreateJS Listeners

- **on**
  
  Shortcut method for `addEventListener`
  Also specify execution scope, only run once, etc...
  ```javascript
  var listener = myBtn.on("click", handleClick, null, false);
  ```

- **off**
  
  Pass returned wrapper function
  ```javascript
  myBtn.on("click", listener);
  ```
Mouse Handler: Player

```javascript
// --- custom method for moving player when mouse moved
player.move = function (mouseEvent) {
    player.x = mouseEvent.stageX - 68;
}

// --- handle mouse movements
player.moveListener = stage.on("stagemousemove", player.move);
```
Player: Add to Stage

```javascript
// --- Add player to stage
stage.addChild(player);
```
Ball Class
Ball Graphics

// BALL GRAPHICS
var ballTossed = new Image();
ballTossed.src = "image/ball.png";

var ballCatchable = new Image();
ballCatchable.src = "image/ball-catch.png";

var ballCaught = new Image();
ballCaught.src = "image/star.gif";
Multiple Game Objects

- Option #1 – CreateJS clone method
- Option #2 – JavaScript class
// BALL CLASS
var Ball = function() {
    var ball = new createjs.Bitmap();
    ball.image = ballTossed;
    ball.x = Math.floor((Math.random() * 600) + 1);
    ball.scaleX = 0.25;
    ball.scaleY = 0.25;
    ball.regX = 50;
    ball.regY = 50;
    ...
}
Ball: Custom Properties

// BALL CLASS
var Ball = function() {

  ...

  // --- custom properties
  ball.state = "Tossed";
  ball.moveToX = Math.floor((Math.random() * 600) + 1);
  ball.moveTime = Math.floor((Math.random() * 2200) + 3200);

  ...

TweenJS

- Implements motion tweens
- Can be chained to create animation sequences
- Can be stacked, but should target different properties with each tween
- Dispatches a change event when a property is changed
- Supports many ease types

http://www.createjs.com/demos/tweenjs/tween_sparktable
TweenJS Example

cREATEJS.TWEEN.get(target)
  .wait(500)
  .to({alpha: 0, visible: false}, 1000)
  .call(handleComplete);

function handleComplete() {
  // Tween complete
}

Ball: Animation

// BALL CLASS
var Ball = function() {
    ...
    // --- move the ball using TweenJS
    createjs.Tween.get(ball)
        .to( { scaleX: 0.75,
             scaleY: 0.75,
             x: ball.moveToX,
             y: 500,
             rotation: 1440 },
        ball.moveTime,
        createjs.Ease.getPowOut(3)
    );
    ...
}
// BALL CLASS
var Ball = function() {
    ...
    // --- custom method for handling mouse clicks on ball
    ball.touched = function () {
        ball.state="Catchable";
        ball.image = ballCatchable
    }
}

// --- handle mouse clicks
ballClickListener = ball.on("mousedown", ball.touched);
...
// BALL CLASS
var Ball = function() {
    ...

    // --- Provide access to the ball object
    return ball;
}

Ball Object Factory
Ball Objects

// BALL OBJECTS
// --- Call addBall method EVERY FRAME

var addBall = createjs.Ticker.on("tick", function addBall () {
    randomNumber = Math.floor((Math.random() * 60) + 1);
    if (randomNumber === 1) {
        stage.addChild(new Ball());
    }
});
Collisions
Custom Event

- **Event Class**
  Create an event string object

- **EventDispatcher Class**
  Dispatch event to listeners
Custom Event example

```javascript
var event = new createjs.Event("progress");
this.dispatchEvent(event);
```
Collision: Player Catch The Ball

```javascript
// BALL CLASS
var Ball = function() {
...

  // --- custom method called when player catches ball
  ball.playerCatchesBall = function() {
    // Change the ball to the caught state
    ball.state = "Caught";
    // Change the ball image to the spinning star
    ball.image = ballCaught;
    // Make the star spin away by moving the registration point
    ball.regX = 130;
    ball.regY = 130;
    // Dispatch event to score a point
    var caughtballEvent = new createjs.Event("caughtball");
    stage.dispatchEvent(caughtballEvent);
  }
};
```
// BALL CLASS
var Ball = function() {

    ...

    // --- custom method called when ball hits player
    ball.ballHitsPlayer = function() {
        // Dispatch event to remove ball click handlers, stop player
        var hitbyballEvent = new createjs.Event("hitbyball");
        stage.dispatchEvent(hitbyballEvent);
    }

    // --- handle ball hitting player, remove handler for clicking ball
    stage.on("hitbyball", function () {
        ball.off("mousedown", ball.clickListener);
    });

};

...
// BALL CLASS
var Ball = function() {
    ...
    // --- custom method for performing hit tests
    ball.hitTest = function() {
        if (this.y > 500) {
            // The ball passed by the player and is off the stage
            stage.removeChild(this);
        } else {
            // Hit Test
            ...
        }
    }
    ...
    ...
}
CreateJS Hit Tests

- **hitTest() method**
  - Test if object intersects a point
  - Must convert point to local coordinates
  - Not very useful, not great performance

- Shape to Shape collisions not supported
  - Can code your own with bounding boxes
  - `getBounds()` method returns rectangle

- **Pythagorean Theorem**
  - Simple hit test approximation
  - Gets distance between two points
// Hit Test
var xDist = this.x - player.x - 70;
var yDist = this.y - player.y - 30;
var distance = Math.sqrt(xDist * xDist + yDist * yDist);

// Using pythagorus
if (distance < 50) {
    if (ball.state === "Catchable") {
        ball.playerCatchesBall();
    } else if (ball.state === "Tossed") {
        ball.ballHitsPlayer();
    }
}
Add Hit Test to Game Loop

```javascript
// BALL CLASS
var Ball = function() {
    ...
    // Call hitTest method EVERY FRAME
    ball.on("tick", ball.hitTest);
    ...
};
```
// SCORE OBJECT
...

// --- custom method for scoring a point when ball caught
score.catch = function (mouseEvent) {
    score.value ++;
    score.text = score.value;
}

// --- handle catching the ball
score.catchListener = stage.on("caughtball", score.catch);
Collision Handler: Player

// PLAYER OBJECT
...

// --- custom method for ending game when player hit by ball
player.hit = function () {
    player.image = playerHit;
    stage.off("stagemousemove", player.moveListener);
}

// --- handle getting hit by a ball
player.hitListener = stage.on("hitbyball", player.hit);
Collision Handler: Balls

// BALL OBJECT FACTORY
...

// --- handle getting hit by a ball, stop the bombardment
stage.on("hitbyball", function () {
    createjs.Ticker.off("tick", addBall);
});
Additional Links

- **CreateJS**
  http://www.createjs.com/

- **Pong Tutorial**
  http://code.tutsplus.com/tutorials/learn-createjs-by-building-an-html5-pong-game--active-11845

- **Atari Arcade**
  https://www.atari.com/arcade/developers

- **Atari Arcade SDK (extends CreateJS)**
  https://github.com/AtariAdmin/AtariArcadeSDK
HTML5 Games with CreateJS

TWITTER: @DAVID_KELLEHER
DAVIDK.NET/GAME