CODE AVENGERS JavaScript 3 Answers

1. Learn about mouse event as you build a basic painting app

1.1. introduce **onMouseDown**

```
function onMouseDown(event) {
  console.log(event.point);
}
```

1.2. introduce **onMouseUp**

```
function onMouseUp(event) {
  var s = new Shape.Circle(event.point, 80);
  s.fillColor = 'blue';
}
```

1.3. introduce **onMouseMove**

```
function onMouseMove(event) {
  var s = new Shape.Circle(event.point, 10);
  s.fillColor = Color.random();
}
```

1.4. introduce **onMouseDrag**

```
var brushColor = 'pink';

function onMouseDrag(event) {
  var s = new Shape.Circle(event.point, 10);
  s.fillColor = brushColor;
  s.opacity = .4;
}

function onMouseUp(event) {
  brushColor = Color.random();
}
```

1.5.

Review Quiz Questions:

1. onMouseDrag
2. onMouseDown
3. onMouseUp
4. onMouseDrag

2. Test some apps and figure out how they work

2.1.

```
function onMouseUp(e) {
  var s = new Shape.Circle(e.point, 30); s.fillColor =
'lightPink';
}

function onMouseMove(e) {
  var s = new Shape.Circle(e.point, 5); s.fillColor =
'lightBlue';
}

function onMouseDown(e) {
  var s = new Shape.Circle(e.point, 40); s.fillColor =
'lightGreen';
}

function onMouseDrag(e) {
  var s = new Shape.Circle(e.point, 10); s.fillColor =
'yellow';
}
```

2.2.

```
var path = new Path();
path.fillColor = 'pink';

function onMouseDown(e) {
  path.selected = true;
  path.add(e);
}

function onMouseUp(e) {
  if(e.event.detail == 2) {
    path.selected = false;
    path = new Path();
    path.fillColor = Color.random();
  }
}

function onMouseDrag(e) {
  path.position.x += e.delta.x;
  path.position.y += e.delta.y;
}

function onMouseMove(e) {
  //This event is not used
}
```

2.3.

```
var r = new Shape.Rectangle();
r.fillColor = 'red';

function onMouseDown(e) {
  r = new Shape.Rectangle(e.point, e.point);
  r.fillColor = 'red';
  r.fillColor.lightness = 0.6;
  r.fillColor.saturation = 0.9;
}

function onMouseUp(e) {
  r.opacity = 0.7;
}

function onMouseDrag(e) {
  r.size.width += e.delta.x * 2;
  r.size.height += e.delta.y * 2;
}

function onMouseMove(e) {
  r.fillColor.hue += e.delta.x;
  r.rotation += e.delta.y;
}
```

2.4.

```
var brush = new Shape.Circle(-10, -10, 10);
brush.fillColor = 'red';

var path;

function onMouseMove(e) {
  brush.position = e.point;
}

function onMouseDown(e) {
  path = new Path();
  path.strokeColor = brush.fillColor;
}

function onMouseDrag(e) {
  brush.position = e.point;
  path.add(e.point);
}

function onMouseUp(e) {
  if(path.length == 0) {
    brush.fillColor.hue += 30;
  }
}
```

2.5.

Review Quiz Questions:

1. s.selected = true
2. fillColor
3. s.rotation += 10
4. s.position.y += 10

3. Use the mouse to create, move and delete shapes

3.1. object Event Handlers

```
var item;

function onMouseDown(e) {
  //Create an ellipse on doubleClick
  if(e.event.detail == 2){
    item = new Shape.Ellipse(e.point, 1, 1);
  } else if (e.event.detail == 1) {
    item = new Shape.Rectangle(e.point, 1, 1);
  } else {
    return;
  }

  item.fillColor = Color.random();
  item.fillColor.lightness = 0.6;
}

function onMouseDrag(e) {
  if (item == null) {
    return;
  }
  //Adjust the size of the item to the current mouse position
  var newBounds = new Rectangle(e.downPoint, e.point);

  if (newBounds.width > 0 && newBounds.height > 0) {
    item.bounds = newBounds;
  }
}

function onMouseUp(e) {
  if (item == null) {
    return;
  } else if (item.bounds.width < 20 || item.bounds.height <
20) {
    //Remove items that are too small
    item.remove();
  }
  item = null;
}
```

3.2. mouse Event Buttons

```
var item;

function onMouseDown(e) {
  if (e.event.button == 2) {
    return;
  }

  //Create an ellipse on doubleClick
  if(e.event.detail == 2){
    item = new Shape.Ellipse(e.point, 1, 1);
  } else if (e.event.detail == 1) {
    item = new Shape.Rectangle(e.point, 1, 1);
  } else {
    return;
  }

  item.fillColor = Color.random();
  item.fillColor.lightness = 0.6;
}

function onMouseDrag(e) {
  if (item == null) {
    return;
  }

  //Adjust the size of the item to the current mouse position
  var newBounds = new Rectangle(e.downPoint, e.point);

  if (newBounds.width > 0 && newBounds.height > 0) {
    item.bounds = newBounds;
  }
}

function onMouseUp(e) {
  if (item == null) {
    return;
  } else if (item.bounds.width < 5 || item.bounds.height < 5)
{
    //Remove items that are too small
    item.remove();
  }
  item = null;
}
```

3.3. item Mouse Events

```
var item;

function onMouseDownItem(e) {
  if (e.event.button == 2 && e.event.detail == 2) {
    this.remove();
  } else if (e.event.detail >= 2) {
    this.fillColor.hue += 30;
  }
}

function onMouseDown(e) {
  if (e.event.button == 2) {
    return;
  }

  //Create an ellipse on doubleClick
  if(e.event.detail == 2){
    item = new Shape.Ellipse(e.point, 1, 1);
  } else if (e.event.detail == 1) {
    item = new Shape.Rectangle(e.point, 1, 1);
  } else {
    return;
  }

  item.onMouseDown = onMouseDownItem;
  item.fillColor = Color.random();
  item.fillColor.lightness = 0.6;
}

function onMouseDrag(e) {
  if (item == null) {
    return;
  }

  //Adjust the size of the item to the current mouse position
  var newBounds = new Rectangle(e.downPoint, e.point);

  if (newBounds.width > 0 && newBounds.height > 0) {
    item.bounds = newBounds;
  }
}

function onMouseUp(e) {
  if (item == null) {
    return;
  } else if (item.bounds.width < 5 || item.bounds.height < 5)
{
    //Remove items that are too small
    item.remove();
  }
  item = null;
}
```

3.4. moving Items

```
var item;

function onMouseDownItem(e) {
  if (e.event.button == 2 && e.event.detail == 2) {
    this.remove();
  } else if (e.event.detail >= 2) {
    this.fillColor.hue += 30;
  }
}

function onMouseDragItem(e) {
  if(e.event.button == 2) {
    this.position.x += e.delta.x;
    this.position.y += e.delta.y;
  }
}

function onMouseDown(e) {
  if (e.event.button == 2) {
    return;
  }

  //Create an ellipse on doubleClick
  if(e.event.detail == 2){
    item = new Shape.Ellipse(e.point, 1, 1);
  } else if (e.event.detail == 1) {
    item = new Shape.Rectangle(e.point, 1, 1);
  } else {
    return;
  }

  item.onMouseDrag = onMouseDragItem;
  item.onMouseDown = onMouseDownItem;
  item.fillColor = Color.random();
  item.fillColor.lightness = 0.6;
}

function onMouseDrag(e) {
  if (item == null) {
    return;
  }

  //Adjust the size of the item to the current mouse position
  var newBounds = new Rectangle(e.downPoint, e.point);

  if (newBounds.width > 0 && newBounds.height > 0) {
    item.bounds = newBounds;
  }
}

function onMouseUp(e) {
  if (item == null) {
    return;
  } else if (item.bounds.width < 5 || item.bounds.height < 5)
{
    //Remove items that are too small
    item.remove();
  }
  item = null;
}
```

3.5.

Review Quiz Questions:

1. circle
2. rectangle
3. square
4. oval

4. Find and fix the bugs in a series of broken apps

4.1.

```
var current = new Path();
current.strokeColor = 'blue';
current.strokeWidth = 5;

function onMouseDown(e) {
  current.add(e.point);

  if(e.event.detail == 2) {
    current = new Path();
    current.strokeColor = Color.random();
  }
}

function onMouseDrag(e) {
  current.add(e.point);
}
```

4.2.

```
var brush = new Shape.Circle(-10,-10, 10);
brush.fillColor = 'blue';
brush.opacity = 0.6;

function onMouseMove(e) {
  brush.position = e.point;
}

function onMouseDrag(e) {
  brush.position = e.point;
  brush.clone();
}

function onMouseDown(e) {
  if (e.event.detail > 1) {
    brush.fillColor.hue += 120;
  }
}
```

4.3.

```
var RADIUS = 10;
var brush = new Shape.Circle(-RADIUS,-RADIUS, RADIUS);
brush.fillColor = 'blue';
brush.opacity = 0.6;

function onMouseMove(x) {
  brush.position = x.point;
}

function onMouseDrag(e) {
  brush.position = e.point;
  brush.clone();
}

function onMouseUp(e) {
  //Toggle the brush shape between circle and ellipse on 2 or
3 clicks
  if (e.event.detail == 2 || e.event.detail == 3) {
    var oldBrush = brush;

    if (brush.shape == 'circle') {
      brush = new Shape.Ellipse(e.point, RADIUS, RADIUS * 2);
    } else if (brush.shape == 'ellipse') {
      brush = new Shape.Circle(e.point, RADIUS);
    }

    //Use the same color and opacity as the previous brush
    brush.fillColor = oldBrush.fillColor;
    brush.opacity = oldBrush.opacity;
    oldBrush.remove();
  }

  //Change the color if 3+ clicks
  if (e.event.detail > 2) {
    brush.fillColor.hue += 30;
  }
}
```

4.4.

```
var item;

//Remove items on triple click
function removeItem(e) {
  if (e.event.detail == 3) {
    this.remove();
  }
}

//Move item on drag
function moveItem(e) {
  if(item != null) return;

  this.position.x += e.delta.x;
  this.position.y += e.delta.y;
}

//Add item on double click
function onMouseDown(e) {
  if (e.event.detail != 2) return;

  item = new Shape.Rectangle(e.point, 1, 1);
  item.fillColor = Color.random();
  item.fillColor.lightness = 0.6;

  item.onMouseDrag = moveItem;
  item.onMouseDown = removeItem;
}

function onMouseDrag(e) {
  if (item == null) return;

  //Adjust the size of the item to the current mouse position
  var newBounds = new Rectangle(e.downPoint, e.point);

  if (newBounds.width > 0 && newBounds.height > 0) {
    item.bounds = newBounds;
  }
}

//Stop drawing previous item when button is released
function onMouseUp(e) {
  if (item == null) {
    return;
  } else if (item.bounds.width < 20 || item.bounds.height <
20) {
    //Remove items that are too small
    item.remove();
  }
  item = null;
}
```

4.5.

5. Make two more apps without any help

5.1. create a Rainbow Brush (part I)

```
//Rainbow Brush
var brush = new Shape.Circle(-10, -10, 10);
brush.fillColor = 'red';

function onMouseMove(e) {
  brush.position = e.point;
}
```

5.2. create a Rainbow Brush (part II)

```
//Rainbow Brush
var brush = new Shape.Circle(-10, -10, 10);
brush.fillColor = 'red';

function onMouseDrag(e) {
  brush.position = e.point;
  brush.clone();
  brush.fillColor.hue += 2;
}

function onMouseMove(e) {
  brush.position = e.point;
  brush.fillColor.hue += 5;
}
```

5.3. create a Polygon App (part I)

```
//Polygon App
var path = new Path();
path.selected = true;

function onMouseDown(e) {
  path.add(e.point);
}
```

5.4. create a Polygon App (part II)

```
var path = new Path();
path.selected = true;

function onMouseDown(e) {
  if (e.event.detail == 1) {
    path.add(e.point);
  } else if(e.event.detail == 2) {
    //Close the previous path
    path.closed = true;
    path.selected = false;
    path.fillColor = new Color.random();
    //Create a new path
    path = new Path();
    path.selected = true;
  }
}
```

5.5.

Review Quiz Questions:

1. hue
2. e.downPoint
3. e.point
4. closed

6. Learn about frame events by bouncing a ball around the screen

6.1.

```
var ball = new Shape.Circle(450, 50, 50);
ball.fillColor = 'red';

function onFrame(event) {
  ball.position.x -= 1;
}
```

6.2.

```
var ball = new Shape.Circle(450, 50, 50);
ball.fillColor = 'red';

var ball2 = new Shape.Circle(50, 50, 50);
ball2.fillColor = 'blue';

function onFrame(event) {
  ball.position.x -= 1;
  ball.position.y += 1;

  ball2.position.x += 1;
  ball2.position.y += 1;
}
```

6.3.

```
var ball = new Shape.Circle(50, 50, 50);
ball.fillColor = 'blue';
ball.speed = 10;

function onFrame(event) {
  ball.position.y += ball.speed;

  if (ball.position.y > 450 || ball.position.y < 50) {
    ball.speed *= -1;
  }
}
```

6.4.

```
var ball = new Shape.Circle(50, 250, 20);
ball.fillColor = 'lightBlue';
ball.speedX = 4;
ball.speedY = 3;

function onFrame(event) {
  ball.position.x += ball.speedX;
  ball.position.y += ball.speedY;

  //Check if the ball hits the bottom or top walls
  if (ball.position.y > 480 || ball.position.y < 20) {
    ball.speedY *= -1;
  }

  //Check if the ball hits the right or left walls
  if (ball.position.x > 480 || ball.position.x < 20) {
    ball.speedX *= -1;
  }
}
```

6.5.

Review Quiz Questions:

1. onFrame
2. 60
3. b.position.x < 20
4. b.speedY *= -1;

7. Make a timing game using animation and mouse events

7.1.

```
//Draw center line
var line = new Path(new Point(250, 0), new Point(250, 500));
line.strokeColor = 'blue';
line.dashArray = [20, 20];

//Create the ball
var radius = 50;
var ball = new Shape.Circle(radius, 250, radius);
ball.fillColor = 'lightBlue';
ball.speed = 3;

//Move the ball
function onFrame(e) {
  ball.position.x += ball.speed;
}
```

7.2.

```
//Draw center line
var line = new Path(new Point(250, 0), new Point(250, 500));
line.strokeColor = 'blue';
line.dashArray = [20, 20];

//Create the ball
var radius = 50;
var ball = new Shape.Circle(radius, 250, radius);
ball.fillColor = 'lightBlue';
ball.speed = 3;

//Move the ball
function onFrame(e) {
  ball.position.x += ball.speed;

  //Bounce off the walls
  if (ball.position.x < radius || ball.position.x > 500 -
radius) {
    ball.speed *= -1;
  }
}
```

7.3.

```
Game.setCursor('/image/game/crosshair.png', 32, 32);
var score = new Game.Score();

//Draw center line
var line = new Path(new Point(250, 0), new Point(250, 500));
line.strokeColor = 'blue';
line.dashArray = [20, 20];

//Create the ball
var radius = 50;
var ball = new Shape.Circle(radius, 250, radius);
ball.fillColor = 'lightBlue';
ball.speed = 3;
ball.onMouseDown = onMouseDownBall;

//Move the ball
function onFrame(e) {
  ball.position.x += ball.speed;

  //Bounce off the walls
  if (ball.position.x < radius || ball.position.x > 500 -
radius) {
    ball.speed *= -1;
  }
}

function onMouseDownBall(e) {
  var distanceFromCenter = Math.abs(250 - ball.position.x);
  score.add(50 - distanceFromCenter);

  //Check if the ball is moving left or right
  if (ball.speed < 0) {
    ball.speed -= 1;
  } else {
    ball.speed += 1;
  }
}
```

7.4.

```
//Create a timer
var timer = new Game.Timer(15);
timer.start();

Game.setCursor('/image/game/crosshair.png', 32, 32);
var score = new Game.Score();

//Draw center line
var line = new Path(new Point(250, 0), new Point(250, 500));
line.strokeColor = 'blue';
line.dashArray = [20, 20];

//Create the ball
var radius = 50;
var ball = new Shape.Circle(radius, 250, radius);
ball.fillColor = 'lightBlue';
ball.speed = 3;
ball.onMouseDown = onMouseDownBall;

//Move the ball
function onFrame(e) {
  if (!timer.isRunning()) return;

  ball.position.x += ball.speed;

  //Bounce off the walls
  if (ball.position.x < radius || ball.position.x > 500 -
radius) {
    ball.speed *= -1;
  }
}

function onMouseDownBall(e) {
  if (!timer.isRunning()) return;

  var distanceFromCenter = Math.abs(250 - ball.position.x);
  score.add(50 - distanceFromCenter);

  //Check if the ball is moving left or right
  if (ball.speed < 0) {
    ball.speed -= 1;
  } else {
    ball.speed += 1;
  }
}
```

7.5.

Review Quiz Questions:

1. 0
2. 600
3. 50
4. -10

8. Take down the satellites in your next home made game

8.1.

```
Game.setBackgroundImage('/image/game/background.jpg');
Game.setCursor('/image/game/crosshair.png', 32, 32);
var score = new Game.Score();
var timer = new Game.Timer(10);
var item = new Raster('/image/game/satellite.png', -100);

var startButton = new Game.Button('Start Game');
startButton.margin = 20;
startButton.onClick = setup;


function setup() {
  for (var i = 0; i < 10; i++) {
    var it = item.clone();
    it.position.x = Game.random();
    it.position.y = Game.random();
  }
  startButton.visible = false;
  timer.start();
}
```

8.2.

```
Game.setBackgroundImage('/image/game/background.jpg');
Game.setCursor('/image/game/crosshair.png', 32, 32);
var score = new Game.Score();
var timer = new Game.Timer(10);
var item = new Raster('/image/game/satellite.png', -100);

var startButton = new Game.Button('Start Game');
startButton.margin = 20;
startButton.onClick = setup;

function setup() {
  for (var i = 0; i < 10; i++) {
    var it = item.clone();
    it.position.x = Game.random();
    it.position.y = Game.random();
    it.rotation = Game.random(360);
    it.onFrame = onFrameItem;
  }
  startButton.visible = false;
  timer.start();
}

function onFrameItem(e) {
  this.position.x += 1;
  this.rotation += 1;
}
```

8.3.

```
Game.setBackgroundImage('/image/game/background.jpg');
Game.setCursor('/image/game/crosshair.png', 32, 32);
var score = new Game.Score();
var timer = new Game.Timer(10);
var item = new Raster('/image/game/satellite.png', -100);

var startButton = new Game.Button('Start Game');
startButton.margin = 20;
startButton.onClick = setup;

function setup() {
  for (var i = 0; i < 10; i++) {
    var it = item.clone();
    it.position.x = Game.random();
    it.position.y = Game.random();
    it.rotation = Game.random(360);
    it.onFrame = onFrameItem;
    it.onMouseDown = onMouseDownItem;
  }
  startButton.visible = false;
  timer.start();
}

function onFrameItem(e) {
  this.position.x += 1;
  this.rotation += 1;
}

function onMouseDownItem(e) {
  if (e.event.detail == 2) {
    this.remove();
    score.add(10);
  }
}
```

8.4.

```
Game.setBackgroundImage('/image/game/background.jpg');
Game.setCursor('/image/game/crosshair.png', 32, 32);
var item = new Raster('/image/game/satellite.png', -100);
var score = new Game.Score();

var timer = new Game.Timer(10);
timer.onTimeout = gameover;

function gameover () {
  startButton.visible = true;
  startButton.text = 'Restart Game';
}

var startButton = new Game.Button('Start Game');
startButton.margin = 20;
startButton.onClick = setup;

function setup() {
  for (var i = 0; i < 10; i++) {
    var it = item.clone();
    it.position.x = Game.random();
    it.position.y = Game.random();
    it.rotation = Game.random(360);
    it.speed = Game.random(0.5, 1);
    it.scaling = it.speed;
    it.onFrame = onFrameItem;
    it.onMouseDown = onMouseDownItem;
  }
  startButton.visible = false;
  timer.restart();
}

function onFrameItem(e) {
  if (!timer.isRunning()) return;

  this.position.x += this.speed;
  this.rotation += this.speed;

  if (this.position.x > 530) {
    this.position.x = -30;
    this.position.y = Game.random();
  }
}

function onMouseDownItem(e) {
  if (!timer.isRunning()) return;

  if (e.event.detail == 2) {
    this.remove();
    score.add(20);
  }
}

function onMouseDown(e) {
  if (e.event.detail == 2) {
    score.add(-10);
  }
}
```

8.5.

Review Quiz Questions:

1. button.visible = false
2. ball.onFrame = moveBall
3. position, rotation
4. item.position.y > 510

9. Fix four more broken animation based apps

9.1.

```
//Create a timer
var timer = new Game.Timer(15);
timer.start();

Game.setCursor('/image/game/crosshair.png', 32, 32);
var score = new Game.Score();

//Draw center line
var line = new Path(new Point(250, 0), new Point(250, 500));
line.strokeColor = 'blue';
line.dashArray = [20, 20];

//Create the ball
var radius = 50;
createBall(100, 150);
createBall(350, 350);

function createBall (x, y) {
  var ball = new Shape.Circle(x, y, radius);
  ball.fillColor = 'lightBlue';
  ball.speed = 4;
  ball.onMouseDown = onMouseDownBall;
  ball.onFrame = onFrameBall;
}

//Move the ball
function onFrameBall(e) {
  if (!timer.isRunning()) return;

  this.position.x += this.speed;

  //Bounce off the walls
  if (this.position.x < radius || this.position.x > 500 -
radius) {
    this.speed *= -1;
    this.fillColor = 'lightBlue';
  }
}

function onMouseDownBall(e) {
  if (!timer.isRunning()) return;

  var distanceFromCenter = Math.abs(250 - this.position.x);
  score.add(50 - distanceFromCenter);
  this.radius -= 4;

  if (distanceFromCenter > 50) {
    this.fillColor = 'red';
  } else {
    this.fillColor = 'lime';
    this.fillColor.hue -= distanceFromCenter * 2.5;
  }
}
```

9.2.

```
var ball = new Shape.Circle(10, 10, 10);
ball.fillColor = 'red';
ball.speed = 9;

//Move balls up and down
function onFrameBall() {
  if (this.position.y < 10 || this.position.y > 490) {
    this.speed *= -1;
  }
  this.position.y += this.speed;
}

function onFrame() {
  var clone = ball.clone();
  clone.speed = 8;
  clone.onFrame = onFrameBall;

  ball.position.x += ball.speed;
  ball.fillColor.hue += -1;

  if (ball.position.x < 10 || ball.position.x > 490) {
    ball.speed *= -1;
  }
}
```

9.3.

```
var p = new Path();
p.strokeColor = 'red';
p.closed = true;
p.strokeWidth = 5;

//Create a random polygon
for (var i = 0; i < 4; i++) {
  var s = p.add(Game.random(), Game.random());
  s.speedX = Game.random(-2, 2);
  s.speedY = Game.random(-2, 2);
}

function onFrame() {
  //Animate each point in the polygon
  for (var i = 0; i < p.segments.length; i++) {
    var s = p.segments[i];
    s.point.x += s.speedX;
    s.point.y += s.speedY;

    if (s.point.x < 0 || s.point.x > 500) {
      s.speedX *= -1;
    }

    if (s.point.y < 0 || s.point.y > 500) {
      s.speedY *= -1;
    }
  }
  p.strokeColor.hue += 1;
}
```

9.4.

```
var base = new Path(new Point(250, 150), new Point(180, 500),
new Point(320, 500));
base.closed = true;
base.fillColor = 'lightGray';
base.strokeColor = 'gray';
base.strokeWidth = 5;
base.strokeJoin = 'round';

var blade = new Path(new Point(250, 25), new Point(230, 100),
new Point(270, 300), new Point(250, 375), new Point(230,
300), new Point(270, 100));
blade.strokeColor = 'yellow';
blade.fillColor = 'lightYellow';
blade.closed = true;
blade.strokeWidth = 5;
blade.smooth();
blade.speed = 10;
blade.onMouseDown = onMouseDownBlade;
blade.onMouseDrag = onMouseDragBlade;

function onFrame() {
  blade.rotation += blade.speed / 4;
  blade.speed *= 0.995;
}

//Slow the blade when clicked
function onMouseDownBlade() {
  this.speed /= 2;
}

//Spin the blade when it is dragged
function onMouseDragBlade(e) {
  if (e.point.y < 200) {
    this.speed = e.delta.x;
  } else {
    this.speed = -e.delta.x;
  }

  if (e.point.x < 200) {
    this.speed -= e.delta.y;
  } else {
    this.speed += e.delta.y;
  }
}
```

9.5.

Review Quiz Questions:

1. 4
2. 6
3. 2
4. 1

10. Get artistic as you create another pair of animation apps

10.1.

```
// Create an animation
function onFrame() {
  var ball = new Shape.Circle(Game.random(), Game.random(),
20);
  ball.fillColor = Color.random();
}
```

10.2.

```
// Create an animation
function onFrame() {
  var ball = new Shape.Circle(Game.random(), Game.random(),
20);
  ball.fillColor = Color.random();
  ball.speed = Game.random(-5, 5);
  ball.onFrame = moveBall;
}

function moveBall(){
  this.position.x += this.speed
  this.position.y += this.speed

  if (this.position.x < 0 || this.position.y < 0
      || this.position.x > 500 || this.position.y > 500) {
    this.speed *= -1;
  }
}
```

10.3.

```
//Create a comet trail
function onMouseDrag(e) {
  var c = new Shape.Circle(e.point, 25);
  c.fillColor = 'orange';
  c.onFrame = shrinkCircle;
}

function shrinkCircle() {
  this.radius -= .5;
  if (this.radius < 1) {
    this.remove();
  }
}
```

10.4.

```
//Create a comet trail
var brushColor;

function onMouseDown() {
  brushColor = Color.random();
  brushColor.lightness = .3;
}

function onMouseDrag(e) {
  var c = new Shape.Circle(e.point, 25);
  c.fillColor = brushColor.clone();
  c.onFrame = shrinkCircle;
}

function shrinkCircle() {
  this.radius -= .5;
  this.fillColor.lightness *= 1.02;
  if (this.radius < 1) {
    this.remove();
  }
}
```

10.5.

Review Quiz Questions:

1. 130
2. top right
3. bottom
4. 180

11. Basic text editor

11.1.

```
var message = new Game.Message('Press a key')

var keyDownCount = 0;
var keyUpCount = 0;

function onKeyDown(e) {
  keyDownCount++;
  message.content = 'onKeyDown ' + keyDownCount;
}

function onKeyUp(e) {
  keyUpCount++;
  message.content = 'onKeyUp ' + keyUpCount;
}
```

11.2.

```
var message = new Game.Message('Press a key');

function onKeyDown(e) {
  message.content = 'onKeyDown\ncharacter: ' + e.character +
'\nkey: ' + e.key;
  if(e.character.length == 0) {
    Game.setBackgroundColor('blue');
  }
}

function onKeyUp(e) {
  message.content = 'onKeyUp\ncharacter: ' + e.character +
'\nkey: ' + e.key;
  Game.setBackgroundColor('');
}
```

11.3.

```
Game.setBackgroundColor('black');
var text = new PointText(20, 40);
text.fillColor = 'white';
text.fontSize = 24;

function onKeyDown(e) {
  console.log(e.key);

  if (e.key == 'tab') {
    text.content += '    ';
  } else if (e.key == 'escape') {
    text.content = '';
  } else if (e.key == 'backspace') {
    var c = text.content;
    text.content = c.substring(0, c.length - 1);
  } else {
    text.content += e.character;

    if (text.bounds.width > 460) {
      var c = text.content;
      text.content = c.substring(0, c.length - 1) + '\n' +
e.character;
    }
  }
}
```

11.4.

```
Game.setBackgroundColor('black');
var marginX = 20;
var marginY = 20;
var fontSize = 24;

var text = new PointText(marginX, fontSize + marginY);
text.fillColor = 'white';
text.fontSize = fontSize

function onKeyDown(e) {
  console.log(e.modifiers);

  if (e.key == 'tab') {
    text.content += '    ';
  } else if (e.key == 'escape') {
    text.content = '';
  } else if (e.key == 'backspace') {
    var c = text.content;
    text.content = c.substring(0, c.length - 1);
  } else {
    text.content += e.character

    if (text.bounds.width > 500 - 2 * marginX) {
      var c = text.content;
      text.content = c.substring(0, c.length - 1) + '\n' +
e.character;
    }
  }

  if (e.modifiers.option) {
    //Change the margins
    if (e.key == 'right') {
      marginX += 10;
      text.point.x += 10;
    } else if (e.key == 'left') {
      marginX -= 10;
      text.point.x -= 10;
    } else if (e.key == 'up') {
      marginY += 10;
      text.point.y += 10;
    } else if (e.key == 'down') {
      marginY -= 10;
      text.point.y -= 10;
    }
  }

  if (e.modifiers.control) {
    //Change the font size
    if (e.key == 'up') {
      text.point.y += 1;
      text.fontSize += 1;
    } else if (e.key == 'down') {
      text.point.y -= 1;
      text.fontSize -= 1;
    }
  }
}
```

11.5.

Review Quiz Questions:

1.
2. option
3. Z
4. a

12. Toggle switch

12.1.

```
var light = new Shape.Circle(250, 250, 150);
light.fillColor = 'red';

var message = new Game.Text('STOP');
message.fontSize = 64;

function onMouseDown(e) {
  toggleVisibleA(light);
  toggleVisibleA(message);
}

function toggleVisibleA(item) {
  if (item.visible == true) {
    item.visible = false;
  } else {
    item.visible = true;
  }
}

function toggleVisibleB(item) {
  if (item.visible) {
    item.visible = false;
  } else {
    item.visible = true;
  }
}
```

12.2.

```
var light = new Shape.Circle(250, 250, 150);
light.fillColor = 'red';

var message = new Game.Text('STOP');
message.fontSize = 64;

function onMouseDown(e) {
  if (!e.item) {
    toggleVisible(light);
    toggleVisible(message);
  }
}

function toggleVisible(item) {
  item.visible = !item.visible;
}
```

12.3.

```
var light = new Shape.Circle(250, 250, 150);
light.fillColor = 'red';

var message = new Game.Text('STOP');
message.fontSize = 64;

function onMouseDown(e) {
  toggleText();
  toggleColor();
  console.log(light.fillColor);
}

function toggleColor() {
  if (light.fillColor == new Color('red')) {
    light.fillColor = 'green';
  } else {
    light.fillColor = 'red';
  }
}

function toggleText() {
  if (message.content == 'STOP') {
    message.content = 'GO';
  } else {
    message.content = 'STOP';
  }
}
```

12.4.

```
Game.setBackgroundColor('yellow');
console.log(Game.getBackgroundColor())

var light = new Shape.Circle(250, 250, 150);
light.fillColor = 'red';

var message = new Game.Text('STOP');
message.fontSize = 64;

function onMouseDown(e) {
  if (e.item) {
    toggleColor();
    toggleText();
  } else {
    toggleBackground();
  }
}

function toggleColor() {
  if (light.fillColor == new Color('red')) {
    light.fillColor = 'green';
  } else {
    light.fillColor = 'red';
  }
}

function toggleText() {
  if (message.content == 'STOP') {
    message.content = 'GO';
  } else {
    message.content = 'STOP';
  }
}

function toggleBackground() {
  if (Game.getBackgroundColor() == 'yellow') {
    Game.setBackgroundColor('black');
  } else {
    Game.setBackgroundColor('yellow');
  }
}
```

12.5.

Review Quiz Questions:

1. 4
2. 5
3. 1
4. 2

13. Typing tutor

13.1.

```
var text = new Game.Text('The quick brown fox jumps over the
lazy dog.', 10, 230)
text.fontSize = 40;

function onKeyDown(e) {
  if (text.content.charAt(0) == e.character) {
    text.content = text.content.substring(1);
  }
}
```

13.2.

```
var text = new Game.Text('The quick brown fox jumps over the
lazy dog.', 10, 230)
text.fontSize = 40;

function onKeyDown(e) {
  if (text.content.charAt(0) == e.character) {
    text.content = text.content.substring(1);
    //Clear the background if there is still text
    if (text.content) {
      Game.setBackgroundColor('');
    } else {
      Game.setBackgroundColor('lightGreen');
    }
  } else if (e.character) {
    Game.setBackgroundColor('pink');
  }
}
```

13.3.

```
var text = new Game.Text('The quick brown fox jumps over the
lazy dog.', 10, 230)
text.fontSize = 40;

var status = new Game.Status('Accuracy: 100%');

var keyCount = 0;
var correctCount = 0;

function onKeyDown(e) {
  if (text.content.charAt(0) == e.character) {
   correctCount++;
   text.content = text.content.substring(1);
   //Clear the background if there is still text
   if (text.content) {
     Game.setBackgroundColor('');
   } else {
     Game.setBackgroundColor('lightGreen');
   }
  } else if (e.character) {
   Game.setBackgroundColor('pink');
  }

  if (e.character) {
    keyCount++;
    status.content = 'Accuracy: '
      + Math.round(correctCount / keyCount * 100)
      + '%';
  }
}
```

13.4.

```
var text = new Game.Text('The quick brown fox jumps over the
lazy dog.', 10, 230)
text.fontSize = 40;

var timer = new Game.Timer(60);
var status = new Game.Status('Accuracy: 100%\nSpeed: 0wpm');

var keyCount = 0;
var correctCount = 0;

function onKeyDown(e) {
  timer.start();

  if (text.content.charAt(0) == e.character) {
   correctCount++;
   text.content = text.content.substring(1);
   //Clear the background if there is still text
   if (text.content) {
     Game.setBackgroundColor('');
   } else {
     Game.setBackgroundColor('lightGreen');
     timer.stop();
   }
  } else if (e.character) {
   Game.setBackgroundColor('pink');
  }

  if (e.character) {
    keyCount++;
    status.content = 'Accuracy: '
      + Math.round(correctCount / keyCount * 100)
      + '%\nSpeed: '
      + Math.round(correctCount / 5 / timer.getTimeElapsed()
* 60)
      + 'wpm';
  }
}
```

13.5.

```
var text = new Game.Text('The quick brown fox jumps over the
lazy dog.', 10, 230)
text.fontSize = 40;

function onTimeout() {
  Game.setBackgroundColor('red');
}

var timer = new Game.Timer(5);
timer.onTimeout = onTimeout;

var status = new Game.Status('Accuracy: 100%\nSpeed:
0wpm\nScore: 0');

var keyCount = 0;
var correctCount = 0;

function onKeyDown(e) {
  timer.start();
  if (!timer.isRunning()) return;

  if (text.content.charAt(0) == e.character) {
    correctCount++;
    text.content = text.content.substring(1);
    //Clear the background if there is still text
    if (text.content) {
      Game.setBackgroundColor('');
    } else {
      Game.setBackgroundColor('green');
      timer.stop();
    }
  } else if (e.character) {
    Game.setBackgroundColor('pink');
  }

  if (e.character) {
    keyCount++;
    status.content = 'Accuracy: '
      + Math.round(correctCount / keyCount * 100)
      + '%\nSpeed: '
      + Math.round(correctCount / 5 / timer.getTimeElapsed()
* 60)
      + 'wpm\nScore: ' + correctCount;
  }
}
```

14. Traffic light

14.1.

```
Game.setBackgroundColor('black')

var light = new Shape.Circle(250, 250, 150);
light.fillColor = 'red';

var message = new Game.Text('STOP');
message.fontSize = 64;

function onMouseDown(e) {
  if (light.fillColor == new Color('red')) {
    light.fillColor = 'yellow';
    message.content = 'READY';
  } else if (light.fillColor == new Color('yellow')) {
    light.fillColor = 'green';
    message.content = 'GO';
  } else if (light.fillColor == new Color('green')) {
    light.fillColor = 'orange';
    message.content = 'SLOW';
  } else {
    light.fillColor = 'red';
    message.content = 'STOP';
  }
}
```

14.2.

```
Game.setBackgroundColor('black')

var light = new Shape.Circle(250, 250, 150);
light.fillColor = 'red';

var message = new Game.Text('STOP');
message.fontSize = 64;

var TEXT = ['STOP', 'READY', 'GO', 'SLOW'];
var COLORS = ['red', 'yellow', 'green', 'orange'];
var currentIndex = 0;

function onMouseDown(e) {
  currentIndex++;
  if (currentIndex >= COLORS.length) {
    currentIndex = 0;
  }
  light.fillColor = COLORS[currentIndex];
  message.content = TEXT[currentIndex];
}
```

14.3.

```
Game.setBackgroundColor('black')

var light = new Shape.Circle(250, 250, 150);
light.fillColor = 'red';

var message = new Game.Text('STOP');
message.fontSize = 64;

var TEXT = ['STOP', 'READY', 'GO', 'SLOW'];
var COLORS = ['red', 'yellow', 'green', 'orange'];
var currentIndex = 0;

function onKeyDown(e) {
  if (e.key == 'right') {
    currentIndex++;
    if (currentIndex >= COLORS.length) {
      currentIndex = 0;
    }
  } else if (e.key == 'left') {
    currentIndex--;
    if (currentIndex < 0) {
      currentIndex = COLORS.length - 1;
    }
  }
  light.fillColor = COLORS[currentIndex];
  message.content = TEXT[currentIndex];
}

function onMouseDown(e) {
  currentIndex--;
  if (currentIndex < 0) {
    currentIndex = COLORS.length - 1;
  }
  light.fillColor = COLORS[currentIndex];
  message.content = TEXT[currentIndex];
}
```

14.4.

14.5.

Review Quiz Questions:

1. items.length - 1
2. currentIndex >= TEXT.length
3. index == TEXT.length - 1
4. Apple

15. Digital whiteboard

15.1.

```
var currentText;
var cursorPosition;

function onMouseMove(e) {
  cursorPosition = e.point;
}

function onKeyDown(e) {
  if (e.character.length > 0) {
    currentText = new PointText(cursorPosition.x,
cursorPosition.y);
    currentText.content = e.character;
    currentText.fontSize = 24;
  }
}
```

15.2.

```
var currentText;
var cursorPosition;

function onMouseMove(e) {
  cursorPosition = e.point;
  currentText  = null;
}

function onKeyDown(e) {
  //Add the character to current item
  if (currentText) {
    if (e.key == 'backspace') {
      var c = currentText.content;
      currentText.content = c.substring(0, c.length - 1);
    } else {
      currentText.content += e.character;
    }
    //Create a new text item
  } else if (e.character) {
    currentText = new PointText(cursorPosition.x,
cursorPosition.y);
    currentText.fillColor = 'black';
    currentText.content = e.character;
    currentText.fontSize = 24;
  }
}
```

15.3.

```
var currentText;
var cursorPosition;

function onMouseMove(e) {
  currentText = e.item;
  cursorPosition = e.point;
}

function onKeyDown(e) {
  //Add the character to current item
  if (currentText) {
    if (e.key == 'delete') {
      currentText.remove();
      currentText = null;
    } else if (e.key == 'backspace') {
      var c = currentText.content;
      currentText.content = c.substring(0, c.length - 1);
    } else {
      currentText.content += e.character;
      currentText.strokeWidth = 0;
    }
    //Create a new text item
  } else if (e.character) {
    currentText = new PointText(cursorPosition.x,
cursorPosition.y);
    currentText.fillColor = 'black';
    currentText.content = e.character;
    currentText.fontSize = 24;
    currentText.onMouseEnter = onEnterItem
    currentText.onMouseLeave = onLeaveItem
  }
}

function onEnterItem(e) {
  currentText = this;
  this.strokeWidth = 1;
  this.strokeColor = 'lightBlue';
}

function onLeaveItem(e) {
  currentText = null;
  this.strokeWidth = 0;
}
```

15.4.

```
var pen = new PointText(10, 28);
pen.content = 'PEN';
pen.fillColor = 'black';
pen.visible = true;
pen.fontSize = 24;

var COLORS = ['black', 'blue', 'red', 'green'];
var currentColor = 0;

var currentText;
var cursorPosition;

function onMouseMove(e) {
  currentText = e.item;
  cursorPosition = e.point;
}

function onKeyDown(e) {
  //Add the character to current item
  if (currentText) {
    if (e.key == 'delete') {
      currentText.remove();
      currentText = null;
    } else if (e.key == 'backspace') {
      var c = currentText.content;
      currentText.content = c.substring(0, c.length - 1);
    } else {
      currentText.content += e.character;
      currentText.strokeWidth = 0;
    }
    //Create a new text item
  } else if (e.character.length > 0) {
    currentText = pen.clone();
    currentText.point = cursorPosition;
    currentText.content = e.character;
    currentText.onMouseEnter = onEnterItem
    currentText.onMouseLeave = onLeaveItem
  }

  if (e.key == 'right') {
    if (++currentColor >= COLORS.length) currentColor = 0;
    pen.fillColor = COLORS[currentColor];
  } else if (e.key == 'left') {
    if (--currentColor < 0) currentColor = COLORS.length - 1;
    pen.fillColor = COLORS[currentColor];
  }
}

function onEnterItem(e) {
  currentText = this;
  this.strokeWidth = 1;
  this.strokeColor = 'lightBlue';
}

function onLeaveItem(e) {
  currentText = null;
  this.strokeWidth = 0;
}
```

15.5.

```
var pen;

function reset() {
  pen = new PointText(10, 28);
  pen.content = 'PEN';
  pen.fillColor = 'black';
  pen.visible = true;
  pen.fontSize = 24;
}
reset();

var COLORS = ['black', 'blue', 'red', 'green'];
var currentColor = 0;

var currentText;
var cursorPosition;

function onMouseMove(e) {
  currentText = e.item;
  cursorPosition = e.point;
}

function onKeyDown(e) {
  //Add the character to current item
  if (currentText) {
    if (e.key == 'delete') {
      currentText.remove();
      currentText = null;
    } else if (e.key == 'backspace') {
      var c = currentText.content;
      currentText.content = c.substring(0, c.length - 1);
    } else {
      currentText.content += e.character;
      currentText.strokeWidth = 0;
    }
    //Create a new text item
  } else if (e.character.length > 0) {
    currentText = pen.clone();
    currentText.point = cursorPosition;
    currentText.content = e.character;
    currentText.onMouseEnter = onEnterItem
    currentText.onMouseLeave = onLeaveItem
  }

  if (e.modifiers.option) {
    if (e.key == 'right') {
      pen.fillColor.hue += 30;
    } else if (e.key == 'left') {
      pen.fillColor.hue += 30;
    } else if (e.key == 'up' && pen.fillColor.saturation < 1)
{
      pen.fillColor.saturation += .2;
    } else if (e.key == 'down' && pen.fillColor.saturation >
0) {
      pen.fillColor.saturation -= .2;
    }
  } else {
    if (e.key == 'right') {
      if (++currentColor >= COLORS.length) currentColor = 0;
      pen.fillColor = COLORS[currentColor];
    } else if (e.key == 'left') {
      if (--currentColor < 0) currentColor = COLORS.length -
1;
      pen.fillColor = COLORS[currentColor];
    } else if (e.key == 'up' && pen.fillColor.lightness) {
      pen.fillColor.lightness += .2;
    } else if (e.key == 'down' && pen.fillColor.saturation >
0 ) {
      pen.fillColor.saturation -= .2;
    }
  }
}

function onEnterItem(e) {
  currentText = this;
  this.strokeWidth = 1;
  this.strokeColor = 'lightBlue';
}

function onLeaveItem(e) {
  currentText = null;
  this.strokeWidth = 0;
}

function onMouseDown(e) {
  if (e.event.detail === 3) {
    project.activeLayer.removeChildren();
    reset();
  }
}
```

16. Spaceship: moving objects smoothly with Key.isDown

16.1.

```
Game.setBackgroundImage('/image/game/background.jpg');
var player = new Raster('/image/game/ship.png', 250, 420);

function onKeyDown(e) {
  if (e.key == 'left') {
    player.position.x -= 5;
  } else if (e.key == 'right') {
    player.position.x += 5;
  } else if (e.key == 'up') {
    player.position.y -= 10;
  } else if (e.key == 'down') {
    player.position.y += 5;
  }
}
```

16.2.

```
Game.setBackgroundImage('/image/game/background.jpg');
var player = new Raster('/image/game/ship.png', 250, 420);

function onKeyDown(e) {
  if (e.key == 'left') {
    player.position.x -= 5;
    player.rotation = -20;
  } else if (e.key == 'right') {
    player.position.x += 5;
    player.rotation = 20;
  } else if (e.key == 'up') {
    player.position.y -= 10;
    player.rotation = 0;
  } else if (e.key == 'down') {
    player.position.y += 5;
    player.rotation = 0;
  }
}
```

16.3.

```
Game.setBackgroundImage('/image/game/background.jpg');
var player = new Raster('/image/game/ship.png', 250, 420);
var speed = 1.5;

function onFrame(e) {
  var deltaX = 0, deltaY = 0;

  if (Key.isDown('left')) {
    deltaX -= speed;
  }

  if (Key.isDown('right')) {
    deltaX += speed;
  }

  if (Key.isDown('up')) {
    deltaY -= speed * 2;
  }

  if (Key.isDown('down')) {
    deltaY += speed;
  }

  player.position.x += deltaX;
  player.position.y += deltaY;
}
```

16.4.

```
Game.setBackgroundImage('/image/game/background.jpg');
var player = new Raster('/image/game/ship.png', 250, 420);
var speed = 1.5;

function onFrame(e) {
  var deltaX = 0, deltaY = 0, angle = 0, distance = speed;

  if (Key.isDown('left')) {
    angle = -20;
    deltaX -= 1;
  }

  if (Key.isDown('right')) {
    angle = 20;
    deltaX += 1;
  }

  if (Key.isDown('up')) {
    deltaY -= 1;
    distance = 2 * speed;
  }

  if (Key.isDown('down')) {
    deltaY += 1;
  }

  var delta = new Point(deltaX, deltaY);
  if (delta.length) {
    delta.length = distance;
    player.position += delta;
  }

  player.rotation = angle;
}
```

16.5.

```
Game.setBackgroundImage('/image/game/background.jpg');
var player = new Raster('/image/game/ship.png', 250, 420);
var speed = 1.5;

function onFrame(e) {
  var deltaX = 0, deltaY = 0, angle = 0, distance = speed;

  if (Key.isDown('left')) {
    angle = -20;
    deltaX -= 1;
  }

  if (Key.isDown('right')) {
    angle = 20;
    deltaX += 1;
  }

  if (Key.isDown('up')) {
    deltaY -= 1;
    distance = 2 * speed;
  }

  if (Key.isDown('down')) {
    deltaY += 1;
  }

  var delta = new Point(deltaX, deltaY);
  if (delta.length) {
    delta.length = distance;
    player.position += delta;
  }

  if (player.rotation > angle) {
    player.rotation--;
  } else if (player.rotation < angle) {
    player.rotation++;
  }
}
```

17. Rotate the hands on a clock

17.1.

```
var path = new Path(new Point(0, 0), new Point(250, 250));
path.strokeColor = 'red';
path.strokeWidth = 4;

var point = path.lastSegment.point;

function onFrame() {
        if (Key.isDown('up')) {
          point.length += 5;
        } else if (Key.isDown('down')) {
          point.length -= 5;
        }

        if (Key.isDown('left')) {
          point.angle -= 1;
        } else if (Key.isDown('right')) {
          point.angle += 1;
        }
}
```

17.2.

```
var centerPoint = new Point(250, 250);

var minute = new Path(centerPoint, new Point(250, 400));
minute.strokeColor = 'black';
minute.strokeWidth = 4;
minute.pivot = centerPoint;

var center = new Shape.Circle(centerPoint, 15);
center.fillColor = 'gold';

var clock = new Shape.Circle(centerPoint, 210);
clock.strokeColor = 'lightBlue';
clock.strokeWidth = 40;

function onFrame() {
        if (Key.isDown('left')) {
          minute.rotation -= 2;
        } else if (Key.isDown('right')) {
          minute.rotation += 2;
        }
}
```

17.3.

```
var centerPoint = new Point(250, 250);

var hour = new Path(centerPoint, new Point(250, 150));
hour.strokeColor = 'black';
hour.strokeWidth = 8;
hour.pivot = centerPoint;

var minute = new Path(centerPoint, new Point(250, 100));
minute.strokeColor = 'black';
minute.strokeWidth = 4;
minute.pivot = centerPoint;

var center = new Shape.Circle(centerPoint, 15);
center.fillColor = 'gold';

var clock = new Shape.Circle(centerPoint, 210);
clock.strokeColor = 'lightBlue';
clock.strokeWidth = 40;

function onFrame() {
        if (Key.isDown('left')) {
          minute.rotation -= 2;
        } else if (Key.isDown('right')) {
          minute.rotation += 2;
        }
}
```

17.4.

```
var centerPoint = new Point(250, 250);

var hour = new Path(centerPoint, new Point(250, 150));
hour.strokeColor = 'black';
hour.strokeWidth = 8;
hour.pivot = centerPoint;

var minute = new Path(centerPoint, new Point(250, 100));
minute.strokeColor = 'black';
minute.strokeWidth = 4;
minute.pivot = centerPoint;

var center = new Shape.Circle(centerPoint, 15);
center.fillColor = 'gold';

var clock = new Shape.Circle(centerPoint, 210);
clock.strokeColor = 'lightBlue';
clock.strokeWidth = 40;

var MINUTE_SPEED = 2;
var HOUR_SPEED = MINUTE_SPEED / 12;

function onFrame() {
        if (Key.isDown('left')) {
          minute.rotation -= MINUTE_SPEED;
          hour.rotation -= HOUR_SPEED;
        } else if (Key.isDown('right')) {
          minute.rotation += MINUTE_SPEED;
          hour.rotation += HOUR_SPEED;
        }
}
```

17.5.

Review Quiz Questions:

1. [200,200]
2. [100,100]
3. [0,200]
4. [400,200]

18. Do collision detection using HitTest

18.1.

```
Game.setBackgroundImage('/image/game/background.jpg');
var player = new Raster('/image/game/ship.png', 250, 420);
var speed = 1.5;

var bomb = new Raster('/image/game/bomb.png', 250, -40);
bomb.onFrame = moveBomb;

function moveBomb () {
  this.position.y += 2;
}

function onFrame(e) {
  var deltaX = 0, deltaY = 0, angle = 0, distance = speed;

  if (Key.isDown('left')) {
    angle = -20;
    deltaX -= 1;
  }

  if (Key.isDown('right')) {
    angle = 20;
    deltaX += 1;
  }

  if (Key.isDown('up')) {
    deltaY -= 1;
    distance = 2 * speed;
  }

  if (Key.isDown('down')) {
    deltaY += 1;
  }

  var delta = new Point(deltaX, deltaY);
  if (delta.length) {
    delta.length = distance;
    player.position += delta;
  }

  if (player.rotation > angle) {
    player.rotation--;
  } else if (player.rotation < angle) {
    player.rotation++;
  }
}
```

18.2.

```
Game.setBackgroundColor('lime');
var player = new Raster('/image/game/ship.png', 250, 420);
var speed = 1.5;

var bomb = new Raster('/image/game/bomb.png', 250, 250);

function onFrame(e) {
  var deltaX = 0, deltaY = 0, angle = 0, distance = speed;

  if (Key.isDown('left')) {
    angle = -20;
    deltaX -= 1;
  }

  if (Key.isDown('right')) {
    angle = 20;
    deltaX += 1;
  }

  if (Key.isDown('up')) {
    deltaY -= 1;
    distance = 2 * speed;
  }

  if (Key.isDown('down')) {
    deltaY += 1;
  }

  var delta = new Point(deltaX, deltaY);
  if (delta.length) {
    delta.length = distance;
    player.position += delta;

    var result = player.hitTest(bomb.position);
    if (result) {
      console.log(result);
      Game.setBackgroundColor('red');
    } else {
      Game.setBackgroundColor('lime');
    }
  }

  if (player.rotation > angle) {
    player.rotation--;
  } else if (player.rotation < angle) {
    player.rotation++;
  }
}
```

18.3.

```
Game.setBackgroundColor('lime');
var player = new Raster('/image/game/ship.png', 250, 420);
var speed = 1.5;

var bomb = new Raster('/image/game/bomb.png', 250, 250);

function onFrame(e) {
  if (player.exploded) return;

  var deltaX = 0, deltaY = 0, angle = 0, distance = speed;

  if (Key.isDown('left')) {
    angle = -20;
    deltaX -= 1;
  }

  if (Key.isDown('right')) {
    angle = 20;
    deltaX += 1;
  }

  if (Key.isDown('up')) {
    deltaY -= 1;
    distance = 2 * speed;
  }

  if (Key.isDown('down')) {
    deltaY += 1;
  }

  var delta = new Point(deltaX, deltaY);
  if (delta.length) {
    delta.length = distance;
    player.position += delta;

    var result = player.hitTest(bomb.position);
    if (result && result.color.alpha) {
      player.exploded = true;
      player.source = '/image/game/explosion1.png';
      bomb.remove();
      Game.setBackgroundColor('red');
    } else {
      Game.setBackgroundColor('lime');
    }
  }

  if (player.rotation > angle) {
    player.rotation--;
  } else if (player.rotation < angle) {
    player.rotation++;
  }
}
```

18.4.

```
Game.setBackgroundImage('/image/game/background.jpg');
var player = new Raster('/image/game/ship.png', 250, 420);
var speed = 2.5;
var bombSpeed = 2;

var bomb = new Raster('/image/game/bomb.png', 250, -40);
bomb.onFrame = moveBomb;

function moveBomb() {
  this.position.y += bombSpeed;

  var result = player.hitTest(this.position);

  if (result && result.color.alpha) {
    this.remove();
    player.exploded = true;
    player.source = '/image/game/explosion1.png';
  }

  if (this.position.y > 520) {
    this.remove();
    bomb = new Raster('/image/game/bomb.png', Game.random(),
-40);
    bomb.onFrame = moveBomb;
    bombSpeed += 0.5;
  }
}

function onFrame(e) {
  if (player.exploded) return;

  var deltaX = 0, deltaY = 0, angle = 0, distance = speed;

  if (Key.isDown('left')) {
    angle = -20;
    deltaX -= 1;
  }

  if (Key.isDown('right')) {
    angle = 20;
    deltaX += 1;
  }

  if (Key.isDown('up')) {
    deltaY -= 1;
    distance = 2 * speed;
  }

  if (Key.isDown('down')) {
    deltaY += 1;
  }

  var delta = new Point(deltaX, deltaY);
  if (delta.length) {
    delta.length = distance;
    player.position += delta;
  }

  if (player.rotation > angle) {
    player.rotation--;
  } else if (player.rotation < angle) {
    player.rotation++;
  }
}
```

18.5.

```
Game.setBackgroundImage('/image/game/background.jpg');
var score = new Game.Score();

var player = new Raster('/image/game/ship.png', 250, 420);
var speed = 2.5;
var bombSpeed = 2;

var bomb = new Raster('/image/game/bomb.png', 250, -40);
bomb.onFrame = moveBomb;

function moveBomb() {
  this.position.y += bombSpeed;

  var result = player.hitTest(this.position);

  if (result && result.color.alpha) {
    this.remove();
    player.exploded = true;
    player.source = '/image/game/explosion1.png';
  }

  if (this.position.y > 520) {
    score.add(1);
    this.remove();
    bomb = new Raster('/image/game/bomb.png', Game.random(),
-40);
    bomb.onFrame = moveBomb;
    bombSpeed += 0.5;
  }
}

function onFrame(e) {
  if (player.exploded) return;

  var deltaX = 0, deltaY = 0, angle = 0, distance = speed;

  if (Key.isDown('left')) {
    angle = -20;
    deltaX -= 1;
  }

  if (Key.isDown('right')) {
    angle = 20;
    deltaX += 1;
  }

  if (Key.isDown('up')) {
    deltaY -= 1;
    distance = 2 * speed;
  }

  if (Key.isDown('down')) {
    deltaY += 1;
  }

  var delta = new Point(deltaX, deltaY);
  if (delta.length) {
    delta.length = distance;
    player.position += delta;
  }

  if (player.rotation > angle) {
    player.rotation--;
  } else if (player.rotation < angle) {
    player.rotation++;
  }
}
```

19. Etch A Sketch: adding vectors

19.1.

```
var v = new Point(-1000, -1000);
var origin = new Point(0, 0);
var v1 = new Point(200, 50);
var v2 = new Point(150, 150);
var v3 = new Point(50, 200);

var line1 = new Path(origin, v1);
line1.strokeColor = 'red';
line1.strokeWidth = 4;

var line2 = new Path(origin, v2);
line2.strokeColor = 'green';
line2.strokeWidth = 4;

var line3 = new Path(origin, v3);
line3.strokeColor = 'blue';
line3.strokeWidth = 4;

var c = new Shape.Circle(origin, 10);
c.fillColor = 'gold'

c.clone();
c.radius *= 1.5;
c.position += v1;

c.clone();
c.radius *= 1.5;
c.position += v2;

c.clone();
c.radius *= 1.5;
c.position += v3;

c.clone();
c.radius *= 1.5;
c.position -= v1;
```

19.2.

```
var v = new Point(-1000, -1000);
var origin = new Point(0, 0);
var v1 = new Point(200, 50);
var v2 = new Point(150, 150);
var v3 = new Point(50, 200);

var line1 = new Path(origin, v1);
line1.strokeColor = 'red';
line1.strokeWidth = 4;

var line2 = new Path(origin, v2);
line2.strokeColor = 'green';
line2.strokeWidth = 4;

var line3 = new Path(origin, v3);
line3.strokeColor = 'blue';
line3.strokeWidth = 4;

var c = new Shape.Circle(origin, 10);
c.fillColor = 'gold'

c.clone();
c.radius *= 1.5;
c.position += v3 / 2;

c.clone();
c.radius *= 1.5;
c.position += v1 * 2;

c.clone();
c.radius *= 1.5;
c.position -= v1 * 1.5;

c.clone();
c.radius *= 1.5;
c.position += v3 / 1.5;
```

19.3.

```
var frame = new Shape.Rectangle(0, 0, 500, 400);
frame.fillColor = 'red';
frame.strokeColor = 'black';
frame.radius = 8;

var screen = new Shape.Rectangle(60, 60, 380, 260);
screen.fillColor = 'whiteSmoke';
screen.radius = 16;
screen.strokeColor = 'black';

var knob1 = new Shape.Circle(35, 360, 30);
knob1.fillColor = 'white';
knob1.strokeColor = 'black';
var knob2 = knob1.clone();
knob2.position.x = 465;

var text = new PointText(new Point(250, 40));
text.width = 500;
text.justification = 'center';
text.fillColor = 'gold';
text.strokeColor = 'black'
text.fontWeight = 'bold'
text.fontFamily = 'cursive';
text.fontSize = 28;
text.content = 'Etch A Sketch';

// The amount to move when keys are pressed
var step = 1;

// The starting position of the line
var lastPoint = new Point(250, 200);

var path = new Path();
path.strokeColor = 'black';
path.add(lastPoint);


function onFrame(event) {
        var delta = new Point(0, 0);

        if(Key.isDown('left')) {
                delta.x -= step;
        }

        if(Key.isDown('right')) {
                delta.x += step;
        }

        if(Key.isDown('up')) {
                delta.y -= step;
        }

        if(Key.isDown('down')) {
                delta.y += step;
        }

        if (delta.length) {
          lastPoint += delta;
          path.add(lastPoint);
        }
}
```

19.4.

```
var border = new Shape.Rectangle(0, 0, 500, 400);
border.fillColor = 'red';
border.strokeColor = 'black';
border.radius = 8;

var screen = new Shape.Rectangle(60, 60, 380, 260);
screen.fillColor = 'whiteSmoke';
screen.radius = 16;
screen.strokeColor = 'black';

var knob1 = new Shape.Circle(35, 360, 30);
knob1.fillColor = 'white';
knob1.strokeColor = 'black';
var knob2 = knob1.clone();
knob2.position.x = 465;

var text = new PointText(new Point(250, 40));
text.width = 500;
text.justification = 'center';
text.fillColor = 'gold';
text.strokeColor = 'black'
text.fontWeight = 'bold'
text.fontFamily = 'cursive';
text.fontSize = 28;
text.content = 'Etch A Sketch';

// The amount we will move when one of the keys is pressed:
var step = 1;

// The starting position of the line
var lastPoint = new Point(250, 200);

var path = new Path();
path.strokeColor = 'black';
path.add(lastPoint);

function onFrame(event) {
        var delta = new Point(0, 0);

        if(Key.isDown('left')) {
                delta.x -= step;
        }

        if(Key.isDown('right')) {
                delta.x += step;
        }

        if(Key.isDown('up')) {
                delta.y -= step;
        }

        if(Key.isDown('down')) {
                delta.y += step;
        }

        if (delta.length) {
          lastPoint += delta;

          if (lastPoint.x < 60 || lastPoint.x > 440
              || lastPoint.y < 60 || lastPoint.y > 320) {
            lastPoint -= delta;
          } else {
            path.add(lastPoint);
          }
        }

        if(Key.isDown('escape')) {
                path.removeSegments();
        }
}
```

19.5.

Review Quiz Questions:

1. [200, 200]
2. [200, 200]
3. [100, 200]
4. [400, 200]

20. Mini golf: practice vector geometry

20.1.

```
Game.setBackgroundImage('/image/game/minigolf1.jpg');

var GAME_SIZE = 500;
var BALL_RADIUS = 4;
var HOLE_RADIUS = 10;

var score = new Game.Score('Strokes')
var message = new Game.Message('Mini Golf');

var startLine = new Path(new Point(20, 100), new Point(170,
100));
startLine.strokeColor = 'orange';
startLine.strokeWidth = 10;

var border = new Path(
  new Point(20, 20),
  new Point(20, 480),
  new Point(480, 480),
  new Point(480, 330),
  new Point(170, 330),
  new Point(170, 20)
);

border.strokeWidth = 20;
border.strokeColor = 'silver'
border.closed = true;

var hole = new Shape.Circle(405, 405, HOLE_RADIUS);
hole.fillColor = 'black';

var ballStartPosition = new Point(100, 80);
var ball = new Shape.Circle(ballStartPosition, BALL_RADIUS);
ball.fillColor = 'white';

var complete = false, velocity = new Point(0, 0);

function onMouseDown(e) {
  message.visible = false;
  score.add(1);

  velocity = e.point - ball.position;
  velocity.length = 2;
}

function onFrame() {
  ball.position += velocity;

  if (ball.position.x < BALL_RADIUS || ball.position.x >=
GAME_SIZE - BALL_RADIUS) {
    velocity.x *= -1;
  }

  if (ball.position.y < BALL_RADIUS || ball.position.y >=
GAME_SIZE - BALL_RADIUS) {
    velocity.y *= -1;
  }
}
```

20.2.

```
Game.setBackgroundImage('/image/game/minigolf1.jpg');

var GAME_SIZE = 500;
var BALL_RADIUS = 4;
var HOLE_RADIUS = 10;

var score = new Game.Score('Strokes')
var message = new Game.Message('Mini Golf');

var startLine = new Path(new Point(20, 100), new Point(170,
100));
startLine.strokeColor = 'orange';
startLine.strokeWidth = 10;

var border = new Path(
  new Point(20, 20),
  new Point(20, 480),
  new Point(480, 480),
  new Point(480, 330),
  new Point(170, 330),
  new Point(170, 20)
);

border.strokeWidth = 20;
border.strokeColor = 'silver'
border.closed = true;

var hole = new Shape.Circle(405, 405, HOLE_RADIUS);
hole.fillColor = 'black';

var ballStartPosition = new Point(100, 80);
var ball = new Shape.Circle(ballStartPosition, BALL_RADIUS);
ball.fillColor = 'white';

var complete = false, velocity = new Point(0, 0);

function onMouseDown(e) {
  message.visible = false;
  score.add(1);

  velocity = e.point - ball.position;
  velocity.length = 2;
}

function onFrame() {
  var newPosition = ball.position + velocity;

  //Check if ball hit wall
  if (border.hitTest(newPosition)) {
    //Check if it hits a vertical wall
    var xPosition = ball.position + new Point(velocity.x, 0);
    if (border.hitTest(xPosition)) {
      velocity.x *= -.7;
    }
    //Check if it hits a horizontal wall
    var yPosition = ball.position + new Point(0, velocity.y);
    if (border.hitTest(yPosition)) {
      velocity.y *= -.7;
    }
  }

  ball.position += velocity;
}
```

20.3.

```
Game.setBackgroundImage('/image/game/minigolf1.jpg');

var GAME_SIZE = 500;
var BALL_RADIUS = 4;
var HOLE_RADIUS = 10;

var score = new Game.Score('Strokes')
var message = new Game.Message('Mini Golf');

var startLine = new Path(new Point(20, 100), new Point(170,
100));
startLine.strokeColor = 'orange';
startLine.strokeWidth = 10;

var border = new Path(
  new Point(20, 20),
  new Point(20, 480),
  new Point(480, 480),
  new Point(480, 330),
  new Point(170, 330),
  new Point(170, 20)
);

border.strokeWidth = 20;
border.strokeColor = 'silver'
border.closed = true;

var hole = new Shape.Circle(405, 405, HOLE_RADIUS);
hole.fillColor = 'black';

var ballStartPosition = new Point(100, 80);
var ball = new Shape.Circle(ballStartPosition, BALL_RADIUS);
ball.fillColor = 'white';

var complete = false, velocity = new Point(0, 0);

function onMouseDown(e) {
  message.visible = false;
  score.add(1);

  velocity = e.point - ball.position;
  velocity.length *= .05;
}

function onFrame() {
  velocity.length *= .992;
  if (velocity.length < .3) velocity.length = 0;

  var newPosition = ball.position + velocity;

  //Check if ball hit wall
  if (border.hitTest(newPosition)) {
    //Check if it hits a vertical wall
    var xPosition = ball.position + new Point(velocity.x, 0);
    if (border.hitTest(xPosition)) {
      velocity.x *= -.95;
    }
    //Check if it hits a horizontal wall
    var yPosition = ball.position + new Point(0, velocity.y);
    if (border.hitTest(yPosition)) {
      velocity.y *= -.95;
    }
  }

  ball.position += velocity;
}
```

20.4.

```
Game.setBackgroundImage('/image/game/minigolf1.jpg');

var GAME_SIZE = 500;
var BALL_RADIUS = 4;
var HOLE_RADIUS = 10;

var score = new Game.Score('Strokes')
var message = new Game.Message('Mini Golf');

var startLine = new Path(new Point(20, 100), new Point(170,
100));
startLine.strokeColor = 'orange';
startLine.strokeWidth = 10;

var border = new Path(
  new Point(20, 20),
  new Point(20, 480),
  new Point(480, 480),
  new Point(480, 330),
  new Point(170, 330),
  new Point(170, 20)
);

border.strokeWidth = 20;
border.strokeColor = 'silver'
border.closed = true;

var hole = new Shape.Circle(405, 405, HOLE_RADIUS);
hole.fillColor = 'black';

var ballStartPosition = new Point(100, 80);
var ball = new Shape.Circle(ballStartPosition, BALL_RADIUS);
ball.fillColor = 'white';

var complete = false, velocity = new Point(0, 0);

function onMouseDown(e) {
  message.visible = false;
  score.add(1);

  velocity = e.point - ball.position;
  velocity.length *= .05;
}

function onFrame() {
  velocity.length *= .992;
  if (velocity.length < .3) velocity.length = 0;

  var newPosition = ball.position + velocity;

  //Check if the ball collides with the HOLE
  if (hole.hitTest(newPosition) && velocity.length < 2) {
    velocity.length = 0;
    ball.position = hole.position;

    message.visible = true;
    if (score.get() == 1) {
      message.content = 'Hole in One!';
    } else {
      message.content = 'Hole complete!';
    }
    return;
  }

  //Check if ball hit wall
  if (border.hitTest(newPosition)) {
    //Check if it hits a vertical wall
    var xPosition = ball.position + new Point(velocity.x, 0);
    if (border.hitTest(xPosition)) {
      velocity.x *= -.95;
    }
    //Check if it hits a horizontal wall
    var yPosition = ball.position + new Point(0, velocity.y);
    if (border.hitTest(yPosition)) {
      velocity.y *= -.95;
    }
  }

  ball.position += velocity;
}
```

20.5.

```
Game.setBackgroundImage('/image/game/minigolf1.jpg');

var GAME_SIZE = 500;
var BALL_RADIUS = 4;
var HOLE_RADIUS = 10;

var score = new Game.Score('Strokes')
var message = new Game.Message('Mini Golf');

var startLine = new Path(new Point(20, 100), new Point(170,
100));
startLine.strokeColor = 'orange';
startLine.strokeWidth = 10;

var border = new Path(
  new Point(20, 20),
  new Point(20, 480),
  new Point(480, 480),
  new Point(480, 330),
  new Point(170, 330),
  new Point(170, 20)
);

border.strokeWidth = 20;
border.strokeColor = 'silver'
border.closed = true;

var hole = new Shape.Circle(405, 405, HOLE_RADIUS);
hole.fillColor = 'black';

var ballStartPosition = new Point(100, 80);
var ball = new Shape.Circle(ballStartPosition, BALL_RADIUS);
ball.fillColor = 'white';

var complete = false, velocity = new Point(0, 0);

function onMouseDown(e) {
  message.visible = false;

  if (complete) {
    complete = false;
    ball.position = ballStartPosition;
    score.reset();
    return;
  }

  score.add(1);

  velocity = e.point - ball.position;
  velocity.length *= .05;
}

function onFrame() {
  velocity.length *= .992;
  if (velocity.length < .3) velocity.length = 0;

  var newPosition = ball.position + velocity;

  //Check if the ball collides with the HOLE
  if (hole.hitTest(newPosition) && velocity.length < 2) {
    complete = true;
    velocity.length = 0;
    ball.position = hole.position;

    message.visible = true;
    if (score.get() == 1) {
      message.content = 'Hole in One!';
    } else {
      message.content = 'Hole complete!';
    }
    return;
  }

  //Check if ball hit wall
  if (border.hitTest(newPosition)) {
    //Check if it hits a vertical wall
    var xPosition = ball.position + new Point(velocity.x, 0);
    if (border.hitTest(xPosition)) {
      velocity.x *= -.95;
    }
    //Check if it hits a horizontal wall
    var yPosition = ball.position + new Point(0, velocity.y);
    if (border.hitTest(yPosition)) {
      velocity.y *= -.95;
```

```
    }
  }

  ball.position += velocity;
}
```

21. Intro to Classes

21.1.

```
var score = 0;

var answer = prompt("Tom's mom hads 4 kids: Nickel, Dime,
Quarter and who?");
if (answer == 'Tom') {
        alert('Correct! 5 more questions...');
        score++;
} else {
  alert('Incorrect!');
}

var answer = prompt('What 5 letter word becomes shorter with
2 letters added on?');
if (answer.toLowerCase() == 'short') {
        alert('Correct! 4 questions to go...');
        score++;
} else {
  alert('Incorrect!');
}

var answer = prompt('A word I know, 6 letters contains.
Subtract just 1, & 12 remains.');
if (answer.toLowerCase() == 'dozens') {
        alert('Correct! 3 to go...');
        score++;
} else {
  alert('Incorrect!');
}

var answer = prompt('What is the next letter: W I T N ...
?');
if (answer == 'L') {
        alert('Correct! 2 to go...');
        score++;
} else {
  alert('Incorrect!');
}

var answer = prompt('What object has keys that open no locks,
space but no room, and you can enter but not go in?');
if (answer.toLowerCase() == 'keyboard') {
        alert('Correct! Last question coming up...');
        score++;
} else {
  alert('Incorrect!');
}

var answer = prompt('A farmer has 5 haystacks in one field
and 4 haystacks in another. How many haystacks would he have
if he combined them all in one field?');
if (answer == '1') {
        alert('Correct! Quiz complete!');
        score++;
} else {
  alert('Incorrect!');
}

alert('You scored ' + score + '/6');
```

21.2.

```
var TOTAL_QUESTIONS = 7;
var currentQuestion = 0;
var score = 0;

function askQuestion(question, answer) {
  currentQuestion++;

  var guess = prompt(question);

  if (guess && guess.toLowerCase() == answer.toLowerCase()) {
    var questionsToGo = TOTAL_QUESTIONS - currentQuestion;
        var message;

        if(questionsToGo == 0) {
          message = 'Quiz Complete!';
        } else if(questionsToGo == 1) {
          message = 'Last question coming up...';
        } else {
          message = questionsToGo + ' questions to go...';
        }

        alert('Correct! ' + message);
        score++;
  } else {
    alert('Incorrect!');
  }
}

askQuestion('Tom\'s mom hads 4 kids: Nickel, Dime, Quarter
and who?', 'Tom');
askQuestion('What 5 letter word becomes shorter with 2
letters added on?', 'short');
askQuestion('A word I know, 6 letters contains. Subtract just
1, & 12 remains.', 'dozens');
askQuestion('What is the next letter: W I T N ... ?', 'L');
askQuestion('What object has keys that open no locks, space
but no room, and you can enter but not go in?', 'keyboard');
askQuestion('A farmer has 5 haystacks in one field and 4
haystacks in another. How many haystacks would he have if he
combined them all in one field?', '1')
askQuestion('Which word in the English language is always
spelled incorrectly?', 'incorrectly');

alert('You scored ' + score + '/' + currentQuestion);
```

21.3.

21.4.

```
var scoreboard = new Game.Score();
scoreboard.add(5);

function addPoint() {
  scoreboard.add(1);
}

var addButton = new Game.Button('+', 260, 0);
var width = addButton.bounds.width;
addButton.onClick = addPoint;

function minusPoint() {
  scoreboard.add(-1);
}

var minusButton = new Game.Button('-', 265 + width, 0);
minusButton.bounds.width = width;
minusButton.onClick = minusPoint;
```

21.5.

Review Quiz Questions:

1. new
2. function
3. Code runs faster
4. instance

22. Tank Duel I: Create a tank class

22.1.

```
function Tank() {
  var tank = new Raster('/image/game/tank1.png');
  return tank;
}

var tank1 = new Tank();
tank1.position = new Point(475, 25);

var tank2 = new Tank();
tank2.position = new Point(25, 475);
```

22.2.

```
function Tank(no) {
  var tank = new Raster('/image/game/tank' + no + '.png');

  if (no == 1) {
    tank.position = new Point(475, 25);
    tank.rotation = 180;
  } else if (no == 2) {
    tank.position = new Point(25, 475);
  }

  return tank;
}

var tank1 = new Tank(1);
var tank2 = new Tank(2);
```

22.3.

```
function Tank(no) {
  //Setup the tank
  var tank = new Raster('/image/game/tank' + no + '.png');

  if (no == 1) {
    tank.position = new Point(475, 25);
    tank.rotation = 180;
  } else if (no == 2) {
    tank.position = new Point(25, 475);
  }

  //Add private variable
  var speed = 2;

  //Add methods
  function move(vector) {
    if (vector.length) {
      vector.length = speed;
      tank.position += vector;
      tank.rotation = vector.angle;
    }
  }
  tank.move = move;

  return tank;
}

var tank1 = new Tank(1);
var tank2 = new Tank(2);

//Handle key presses
function onFrame() {
  var delta = new Point(0, 0);

  if (Key.isDown('left')) {
    delta.x -= 1;
  }

  if (Key.isDown('right')) {
    delta.x += 1;
  }

  if (Key.isDown('up')) {
    delta.y -= 1;
  }

  if (Key.isDown('down')) {
    delta.y += 1;
  }

  tank1.move(delta);
}
```

22.4.

```
function Tank(no) {
  //Setup the tank
  var tank = new Raster('/image/game/tank' + no + '.png');

  if (no == 1) {
    tank.position = new Point(475, 25);
    tank.rotation = 180;
  } else if (no == 2) {
    tank.position = new Point(25, 475);
  }

  //Add private variable
  var speed = 2;

  //Add methods
  tank.move = function(vector) {
    if (vector.length) {
      vector.length = speed;
      tank.position += vector;
      tank.rotation = vector.angle;
    }
  };

  tank.teleport = function() {
    this.position.x = Game.random(25, 475);
    this.position.y = Game.random(25, 475);
  };

  return tank;
}

var tank1 = new Tank(1);
var tank2 = new Tank(2);

//Handle key presses
function onFrame() {
  var delta = new Point(0, 0);

  if (Key.isDown('left')) {
    delta.x -= 1;
  }

  if (Key.isDown('right')) {
    delta.x += 1;
  }

  if (Key.isDown('up')) {
    delta.y -= 1;
  }

  if (Key.isDown('down')) {
    delta.y += 1;
  }

  tank1.move(delta);
}

function onKeyDown(e) {
  if (e.key == 'backspace') {
    tank1.teleport();
  }
}
```

22.5.

```
function Tank(no) {
  //Add private variables
  var tank = new Raster('/image/game/tank' + no + '.png');
  var speed = 2;
  var upKey, downKey, leftKey, rightKey;

  if (no == 1) {
    upKey = 'up', downKey = 'down', leftKey = 'left',
rightKey = 'right';
    tank.position = new Point(475, 25);
    tank.rotation = 180;
  } else if (no == 2) {
    upKey = 'w', downKey = 's', leftKey = 'a', rightKey =
'd';
    tank.position = new Point(25, 475);
  }

  //Add methods
  tank.move = function(vector) {
    if (vector.length) {
      vector.length = speed;
      tank.position += vector;
      tank.rotation = vector.angle;
    }
  };

  tank.teleport = function() {
    this.position.x = Game.random(25, 475);
    this.position.y = Game.random(25, 475);
  };

  //Handle key presses
  tank.onFrame = function() {
    var delta = new Point(0, 0);

    if (Key.isDown(leftKey)) {
      delta.x -= 1;
    }

    if (Key.isDown(rightKey)) {
      delta.x += 1;
    }

    if (Key.isDown(upKey)) {
      delta.y -= 1;
    }

    if (Key.isDown(downKey)) {
      delta.y += 1;
    }

    tank.move(delta);
  };

  return tank;
}

var tank1 = new Tank(1);
var tank2 = new Tank(2);

function onKeyDown(e) {
  if (e.key == 'backspace') {
    tank1.teleport();
  }
}
```

23. Task Duel II: Using the debugger

23.1.

```
var SCREEN_SIZE = 500;

function Tank(no) {
  //Add private variables
  var tank = new Raster('/image/game/tank' + no + '.png');
  var speed = 2;
  var upKey, downKey, leftKey, rightKey;
  var nextShot = 0;

  var RADIUS = 25;

  if (no == 1) {
    upKey = 'up', downKey = 'down', leftKey = 'left',
rightKey = 'right';
    tank.position = new Point(SCREEN_SIZE - RADIUS, RADIUS);
    tank.rotation = 180;
  } else if (no == 2) {
    upKey = 'w', downKey = 's', leftKey = 'a', rightKey =
'd';
    tank.position = new Point(RADIUS, SCREEN_SIZE - RADIUS);
  }

  //Add methods
  tank.move = function(vector) {
    if (vector.length) {
      vector.length = speed;

      var newPosition = tank.position + vector;
      var newY = Math.round(newPosition.y);
      var newX = Math.round(newPosition.x);

      //Stop the tank from going off the screen
      if (newX < RADIUS || newX > SCREEN_SIZE - RADIUS
          || newY < RADIUS || newY > SCREEN_SIZE - RADIUS) {
        return;
      }

      tank.position += vector;
      tank.rotation = vector.angle;
    }
  };

  console.log('Attach onFrame');
  //Handle key presses
  tank.onFrame = function(e) {
    console.log('Run onFrame')

    var delta = new Point(0, 0);

    if (Key.isDown(leftKey)) {
      delta.x -= 1;
    }

    if (Key.isDown(rightKey)) {
      delta.x += 1;
    }

    if (Key.isDown(upKey)) {
      delta.y -= 1;
    }

    if (Key.isDown(downKey)) {
      delta.y += 1;
    }

    tank.move(delta);
  };

  return tank;
}

new Tank(1);
new Tank(2);
```

23.2.

```
var SCREEN_SIZE = 500;

function Bullet(position, rotation) {
  var velocity = new Point(5, 0);
  velocity.angle = rotation;

  var position = position + velocity * 6;
  var source = '/image/game/bullet.png';
  var bullet = new Raster(source, position);

  bullet.onFrame = function() {
    this.position += velocity;
  };

  return bullet;
};

function Tank(no) {
  //Add private variables
  var tank = new Raster('/image/game/tank' + no + '.png');
  var speed = 2;
  var upKey, downKey, leftKey, rightKey, shootKey;
  var nextShot = 0;

  var RADIUS = 25;

  if (no == 1) {
    upKey = 'up', downKey = 'down', leftKey = 'left',
rightKey = 'right', shootKey = 'enter';
    tank.position = new Point(SCREEN_SIZE - RADIUS, RADIUS);
    tank.rotation = 180;
  } else if (no == 2) {
    upKey = 'w', downKey = 's', leftKey = 'a', rightKey =
'd', shootKey = 'tab';
    tank.position = new Point(RADIUS, SCREEN_SIZE - RADIUS);
  }

  //Add methods
  tank.shoot = function() {
    new Bullet(tank.position, tank.rotation);
  };

  tank.move = function(vector) {
    if (vector.length) {
      vector.length = speed;

      var newPosition = tank.position + vector;
      var newY = Math.round(newPosition.y);
      var newX = Math.round(newPosition.x);

      //Stop the tank from going off the screen
      if (newX < RADIUS || newX > SCREEN_SIZE - RADIUS
          || newY < RADIUS || newY > SCREEN_SIZE - RADIUS) {
        return;
      }

      tank.position += vector;
      tank.rotation = vector.angle;
    }
  };

  //Handle key presses
  tank.onFrame = function(e) {
    var delta = new Point(0, 0);

    if (Key.isDown(shootKey)) {
      this.shoot();
    }

    if (Key.isDown(leftKey)) {
      delta.x -= 1;
    }

    if (Key.isDown(rightKey)) {
      delta.x += 1;
    }

    if (Key.isDown(upKey)) {
      delta.y -= 1;
    }

    if (Key.isDown(downKey)) {
      delta.y += 1;
    }

    tank.move(delta);
  };

  return tank;
}

new Tank(1);
new Tank(2);
```

23.3.

```
var SCREEN_SIZE = 500;

function Bullet(position, rotation) {
  var velocity = new Point(5, 0);
  velocity.angle = rotation;

  var position = position + velocity * 6;
  var source = '/image/game/bullet.png';
  var bullet = new Raster(source, position);
  var RADIUS = 5;

  bullet.onFrame = function() {
    this.position += velocity;

    if (this.position.x < RADIUS || this.position.x >
SCREEN_SIZE - RADIUS) {
      velocity.x *= -1;
    }

    if (this.position.y < RADIUS || this.position.y >
SCREEN_SIZE - RADIUS) {
      velocity.y *= -1;
    }

  };

  return bullet;
};

function Tank(no) {
  //Add private variables
  var tank = new Raster('/image/game/tank' + no + '.png');
  var speed = 2;
  var upKey, downKey, leftKey, rightKey, shootKey;
  var nextShot = 0;

  var RADIUS = 25;

  if (no == 1) {
    upKey = 'up', downKey = 'down', leftKey = 'left',
rightKey = 'right', shootKey = 'enter';
    tank.position = new Point(SCREEN_SIZE - RADIUS, RADIUS);
    tank.rotation = 180;
  } else if (no == 2) {
    upKey = 'w', downKey = 's', leftKey = 'a', rightKey =
'd', shootKey = 'tab';
    tank.position = new Point(RADIUS, SCREEN_SIZE - RADIUS);
  }

  //Add methods
  tank.shoot = function() {
    new Bullet(tank.position, tank.rotation);
  };

  tank.move = function(vector) {
    if (vector.length) {
      vector.length = speed;

      var newPosition = tank.position + vector;
      var newY = Math.round(newPosition.y);
      var newX = Math.round(newPosition.x);

      //Stop the tank from going off the screen
      if (newX < RADIUS || newX > SCREEN_SIZE - RADIUS
          || newY < RADIUS || newY > SCREEN_SIZE - RADIUS) {
        return;
      }

      tank.position += vector;
      tank.rotation = vector.angle;
    }
  };

  //Handle key presses
  tank.onFrame = function(e) {
    var delta = new Point(0, 0);

    if (Key.isDown(shootKey) && e.time > nextShot) {
      this.shoot();
      nextShot = e.time + 2;
    }

    if (Key.isDown(leftKey)) {
      delta.x -= 1;
    }

    if (Key.isDown(rightKey)) {
      delta.x += 1;
    }

    if (Key.isDown(upKey)) {
      delta.y -= 1;
    }

    if (Key.isDown(downKey)) {
      delta.y += 1;
    }

    tank.move(delta);
  };

  return tank;
}

new Tank(1);
new Tank(2);
```

23.4.

```
debugger;
var SCREEN_SIZE = 500;
var gameover = false;

function Bullet(position, rotation) {
  var velocity = new Point(5, 0);
  velocity.angle = rotation;

  var position = position + velocity * 6;
  var source = '/image/game/bullet.png';
  var bullet = new Raster(source, position);
  var RADIUS = 5;

  bullet.onFrame = function() {
    this.position += velocity;

    if (this.position.x < RADIUS || this.position.x >
SCREEN_SIZE - RADIUS) {
      velocity.x *= -1;
    }

    if (this.position.y < RADIUS || this.position.y >
SCREEN_SIZE - RADIUS) {
      velocity.y *= -1;
    }

    this.sendToBack();

    //Check if the bullet hit a tank or bullet
    var result = project.activeLayer.hitTest(this.position);

    //Check if it hit another item
    if (result && result.item != this) {
      //Check that they actually hit each other
      if (result.color.alpha == 0) return;

      //Check if the bullet hit another bullet
      if (result.item.source.endsWith(source)) {
        result.item.remove();
      } else {
        result.item.explode();
      }
      this.remove();
    }
  };

  return bullet;
};

function Tank(no) {
  //Add private variables
  var tank = new Raster('/image/game/tank' + no + '.png');
  var speed = 2;
  var upKey, downKey, leftKey, rightKey, shootKey;
  var nextShot = 0;

  var RADIUS = 25;

  if (no == 1) {
    upKey = 'up', downKey = 'down', leftKey = 'left',
rightKey = 'right', shootKey = 'enter';
    tank.position = new Point(SCREEN_SIZE - RADIUS, RADIUS);
    tank.rotation = 180;
  } else if (no == 2) {
    upKey = 'w', downKey = 's', leftKey = 'a', rightKey =
'd', shootKey = 'tab';
    tank.position = new Point(RADIUS, SCREEN_SIZE - RADIUS);
  }

  //Add methods
  tank.explode = function() {
    this.source = '/image/game/explosion.png';
  };

  tank.shoot = function() {
    new Bullet(tank.position, tank.rotation);
  };

  tank.move = function(vector) {
    if (vector.length) {
      vector.length = speed;

      var newPosition = tank.position + vector;
      var newY = Math.round(newPosition.y);
      var newX = Math.round(newPosition.x);

      //Stop the tank from going off the screen
      if (newX < RADIUS || newX > SCREEN_SIZE - RADIUS
          || newY < RADIUS || newY > SCREEN_SIZE - RADIUS) {
        return;
      }

      tank.position += vector;
      tank.rotation = vector.angle;
    }
  };

  //Handle key presses
  tank.onFrame = function(e) {
    var delta = new Point(0, 0);

    if (Key.isDown(shootKey) && e.time > nextShot) {
      this.shoot();
      nextShot = e.time + 2;
    }

    if (Key.isDown(leftKey)) {
      delta.x -= 1;
    }

    if (Key.isDown(rightKey)) {
      delta.x += 1;
    }

    if (Key.isDown(upKey)) {
      delta.y -= 1;
    }

    if (Key.isDown(downKey)) {
      delta.y += 1;
    }

    tank.move(delta);
  };

  return tank;
}

new Tank(1);
new Tank(2);
```

23.5.

```
debugger;
var SCREEN_SIZE = 500;
var gameover = false;

function Bullet(position, rotation) {
  var velocity = new Point(5, 0);
  velocity.angle = rotation;

  var position = position + velocity * 6;
  var source = '/image/game/bullet.png';
  var bullet = new Raster(source, position);
  var RADIUS = 5;

  bullet.onFrame = function() {
    if (gameover) {
      return;
    }

    this.position += velocity;

    if (this.position.x < RADIUS || this.position.x >
SCREEN_SIZE - RADIUS) {
      velocity.x *= -1;
    }

    if (this.position.y < RADIUS || this.position.y >
SCREEN_SIZE - RADIUS) {
      velocity.y *= -1;
    }

    this.sendToBack();

    //Check if the bullet hit a tank or bullet
    var result = project.activeLayer.hitTest(this.position);

    //Check if it hit another item
    if (result && result.item != this) {
      //Check that they actually hit each other
      if (result.color.alpha == 0) return;

      //Check if the bullet hit another bullet
      if (result.item.source.endsWith(source)) {
        result.item.remove();
      } else {
        result.item.explode();
      }
      this.remove();
    }
  };

  return bullet;
};

function Tank(no) {
  //Add private variables
  var tank = new Raster('/image/game/tank' + no + '.png');
  var speed = 2;
  var upKey, downKey, leftKey, rightKey, shootKey;
  var nextShot = 0;

  var RADIUS = 25;

  if (no == 1) {
    upKey = 'up', downKey = 'down', leftKey = 'left',
rightKey = 'right', shootKey = 'enter';
    tank.position = new Point(SCREEN_SIZE - RADIUS, RADIUS);
    tank.rotation = 180;
  } else if (no == 2) {
    upKey = 'w', downKey = 's', leftKey = 'a', rightKey =
'd', shootKey = 'tab';
    tank.position = new Point(RADIUS, SCREEN_SIZE - RADIUS);
  }

  //Add methods
  tank.explode = function() {
    this.source = '/image/game/explosion.png';
    gameover = true;
  };

  tank.shoot = function() {
    new Bullet(tank.position, tank.rotation);
  };

  tank.move = function(vector) {
    if (vector.length) {
      vector.length = speed;

      var newPosition = tank.position + vector;
      var newY = Math.round(newPosition.y);
      var newX = Math.round(newPosition.x);

      //Stop the tank from going off the screen
      if (newX < RADIUS || newX > SCREEN_SIZE - RADIUS
          || newY < RADIUS || newY > SCREEN_SIZE - RADIUS) {
        return;
      }

      tank.position += vector;
      tank.rotation = vector.angle;
    }
  };

  //Handle key presses
  tank.onFrame = function(e) {
    if (gameover) {
      return;
    }

    var delta = new Point(0, 0);

    if (Key.isDown(shootKey) && e.time > nextShot) {
      this.shoot();
      nextShot = e.time + 2;
    }

    if (Key.isDown(leftKey)) {
      delta.x -= 1;
    }

    if (Key.isDown(rightKey)) {
      delta.x += 1;
    }

    if (Key.isDown(upKey)) {
      delta.y -= 1;
    }

    if (Key.isDown(downKey)) {
      delta.y += 1;
    }

    tank.move(delta);
  };

  return tank;
}

Game.setBackgroundImage('/image/game/tankBackground.png');
new Tank(1);
new Tank(2);
```

24. Banking & scoring: understanding classes

24.1.

24.2.

24.3.

24.4.

24.5.

Review Quiz Questions:

1. 1
2. 8
3. 17
4. 2

25. Create classes for a digital scoreboard

25.1.

```
function SSD(origin) {
  var l = 50; //Length
  var w = l / 4; //width

  //Create the seven segments
  var ss = [];
  ss.push(new Shape.Rectangle(origin + new Point(w, 0), l,
w))
  ss.push(new Shape.Rectangle(origin + new Point(l + w, w),
w, l));
  ss.push(new Shape.Rectangle(origin + new Point(l + w, l + w
* 2), w, l));
  ss.push(new Shape.Rectangle(origin + new Point(w, (l + w) *
2), l, w));
  ss.push(new Shape.Rectangle(origin + new Point(0, l + w *
2), w, l));
  ss.push(new Shape.Rectangle(origin + new Point(0, w), w,
l));
  ss.push(new Shape.Rectangle(origin + new Point(w, w + l),
l, w));

  //Set the color of the segments
  for (var i = 0; i < ss.length; i++) {
    ss[i].fillColor = 'red';
  }
}

new SSD(new Point(10, 10));
new SSD(new Point(100, 10));
```

25.2.

```
function SSD(origin) {
  var l = 50; //Length
  var w = l / 4; //width

  //Create the seven segments
  var ss = [];
  ss.push(new Shape.Rectangle(origin + new Point(w, 0), l,
w))
  ss.push(new Shape.Rectangle(origin + new Point(l + w, w),
w, l));
  ss.push(new Shape.Rectangle(origin + new Point(l + w, l + w
* 2), w, l));
  ss.push(new Shape.Rectangle(origin + new Point(w, (l + w) *
2), l, w));
  ss.push(new Shape.Rectangle(origin + new Point(0, l + w *
2), w, l));
  ss.push(new Shape.Rectangle(origin + new Point(0, w), w,
l));
  ss.push(new Shape.Rectangle(origin + new Point(w, w + l),
l, w));

  //Set the color of the segments
  for (var i = 0; i < ss.length; i++) {
    ss[i].fillColor = 'red';
  }

  //Method that sets the number displayed
  this.set = function(n) {
    //Turn them all on
    for (var i = 0; i < ss.length; i++) {
      ss[i].visible = true;
    }

    if (n == 0) {
      ss[6].visible = false;
    } else if (n == 1) {
      ss[0].visible = ss[3].visible = ss[4].visible
        = ss[5].visible = ss[6].visible = false;
    } else if (n == 2) {
      ss[2].visible = ss[5].visible = false;
    } else if (n == 3) {
      ss[4].visible = ss[5].visible = false;
    } else if (n == 4) {
      ss[0].visible = ss[3].visible = ss[4].visible = false;
    } else if (n == 5) {
      ss[1].visible = ss[4].visible = false;
    } else if (n == 6) {
      ss[1].visible = false;
    } else if (n == 7) {
      ss[3].visible = ss[4].visible = ss[5].visible =
ss[6].visible = false;
    } else if (n == 9) {
      ss[4].visible = false;
    }
  }
}


var s0 = new SSD(new Point(10, 10)); s0.set(0);
var s1 = new SSD(new Point(100, 10)); s1.set(1);
var s2 = new SSD(new Point(190, 10)); s2.set(2);
var s3 = new SSD(new Point(280, 10)); s3.set(3);
var s4 = new SSD(new Point(370, 10)); s4.set(4);

var s5 = new SSD(new Point(10, 170)); s5.set(5);
var s6 = new SSD(new Point(100, 170)); s6.set(6);
var s7 = new SSD(new Point(190, 170)); s7.set(7);
var s8 = new SSD(new Point(280, 170)); s8.set(8);
var s9 = new SSD(new Point(370, 170)); s9.set(9);
```

25.3.

```
function SSD(origin) {
  var l = 50; //Length
  var w = l / 4; //width
  var currentNo;

  //Create the seven segments
  var ss = [];
  ss.push(new Shape.Rectangle(origin + new Point(w, 0), l,
w))
  ss.push(new Shape.Rectangle(origin + new Point(l + w, w),
w, l));
  ss.push(new Shape.Rectangle(origin + new Point(l + w, l + w
* 2), w, l));
  ss.push(new Shape.Rectangle(origin + new Point(w, (l + w) *
2), l, w));
  ss.push(new Shape.Rectangle(origin + new Point(0, l + w *
2), w, l));
  ss.push(new Shape.Rectangle(origin + new Point(0, w), w,
l));
  ss.push(new Shape.Rectangle(origin + new Point(w, w + l),
l, w));

  //Set the color of the segments
  for (var i = 0; i < ss.length; i++) {
    ss[i].fillColor = 'red';
  }

  //Method that sets the number displayed
  this.set = function(n) {
    currentNo = n;
    //Turn them all on
    for (var i = 0; i < ss.length; i++) {
      ss[i].visible = true;
    }

    if (n == 0) {
      ss[6].visible = false;
    } else if (n == 1) {
      ss[0].visible = ss[3].visible = ss[4].visible
        = ss[5].visible = ss[6].visible = false;
    } else if (n == 2) {
      ss[2].visible = ss[5].visible = false;
    } else if (n == 3) {
      ss[4].visible = ss[5].visible = false;
    } else if (n == 4) {
      ss[0].visible = ss[3].visible = ss[4].visible = false;
    } else if (n == 5) {
      ss[1].visible = ss[4].visible = false;
    } else if (n == 6) {
      ss[1].visible = false;
    } else if (n == 7) {
      ss[3].visible = ss[4].visible = ss[5].visible =
ss[6].visible = false;
    } else if (n == 9) {
      ss[4].visible = false;
    }
  }

  this.add = function(n) {
    currentNo += n;
    if (currentNo < 0) currentNo += 10;
    currentNo = currentNo % 10;
    this.set(currentNo);
  };

  this.set(0);
}


var score = new SSD(new Point(10, 10));

var addButton = new Game.Button('+', 0, 170);
var minusButton = new Game.Button('-', 50, 170);
minusButton.bounds.width = addButton.bounds.width;

addButton.onMouseDown = function() {
  score.add(1);
};

minusButton.onMouseDown = function() {
  score.add(-1);
};
```

25.4.

```
function SSD(origin) {
  var l = 50; //Length
  var w = l / 4; //width
  var currentNo;

  //Create the seven segments
  var ss = [];
  ss.push(new Shape.Rectangle(origin + new Point(w, 0), l,
w))
  ss.push(new Shape.Rectangle(origin + new Point(l + w, w),
w, l));
  ss.push(new Shape.Rectangle(origin + new Point(l + w, l + w
* 2), w, l));
  ss.push(new Shape.Rectangle(origin + new Point(w, (l + w) *
2), l, w));
  ss.push(new Shape.Rectangle(origin + new Point(0, l + w *
2), w, l));
  ss.push(new Shape.Rectangle(origin + new Point(0, w), w,
l));
  ss.push(new Shape.Rectangle(origin + new Point(w, w + l),
l, w));

  //Set the color of the segments
  for (var i = 0; i < ss.length; i++) {
    ss[i].fillColor = 'red';
  }

  //Method that sets the number displayed
  this.set = function(n) {
    currentNo = n;
    //Turn them all on
    for (var i = 0; i < ss.length; i++) {
      ss[i].visible = true;
    }

    if (n == 0) {
      ss[6].visible = false;
    } else if (n == 1) {
      ss[0].visible = ss[3].visible = ss[4].visible
        = ss[5].visible = ss[6].visible = false;
    } else if (n == 2) {
      ss[2].visible = ss[5].visible = false;
    } else if (n == 3) {
      ss[4].visible = ss[5].visible = false;
    } else if (n == 4) {
      ss[0].visible = ss[3].visible = ss[4].visible = false;
    } else if (n == 5) {
      ss[1].visible = ss[4].visible = false;
    } else if (n == 6) {
      ss[1].visible = false;
    } else if (n == 7) {
      ss[3].visible = ss[4].visible = ss[5].visible =
ss[6].visible = false;
    } else if (n == 9) {
      ss[4].visible = false;
    }
  }

  this.add = function(n) {
    currentNo += n;
    currentNo = currentNo % 10;
    this.set(currentNo);
  };

  this.set(0);
}

function Score(origin) {
  var digit1 = new SSD(origin);
  var digit2 = new SSD(origin + new Point(100, 0));
}

var score = new Score(new Point(10, 10));

var addButton = new Game.Button('+', 0, 170);
var minusButton = new Game.Button('-', 50, 170);
minusButton.bounds.width = addButton.bounds.width;

addButton.onMouseDown = function() {
  score.add(1);
};

minusButton.onMouseDown = function() {
```

```
  score.add(-1);
};
```

25.5.

```
function SSD(origin) {
  var l = 50; //Length
  var w = l / 4; //width
  var currentNo;

  //Create the seven segments
  var ss = [];
  ss.push(new Shape.Rectangle(origin + new Point(w, 0), l,
w))
  ss.push(new Shape.Rectangle(origin + new Point(l + w, w),
w, l));
  ss.push(new Shape.Rectangle(origin + new Point(l + w, l + w
* 2), w, l));
  ss.push(new Shape.Rectangle(origin + new Point(w, (l + w) *
2), l, w));
  ss.push(new Shape.Rectangle(origin + new Point(0, l + w *
2), w, l));
  ss.push(new Shape.Rectangle(origin + new Point(0, w), w,
l));
  ss.push(new Shape.Rectangle(origin + new Point(w, w + l),
l, w));

  //Set the color of the segments
  for (var i = 0; i < ss.length; i++) {
    ss[i].fillColor = 'red';
  }

  //Method that sets the number displayed
  this.set = function(n) {
    currentNo = n;
    //Turn them all on
    for (var i = 0; i < ss.length; i++) {
      ss[i].visible = true;
    }

    if (n == 0) {
      ss[6].visible = false;
    } else if (n == 1) {
      ss[0].visible = ss[3].visible = ss[4].visible
        = ss[5].visible = ss[6].visible = false;
    } else if (n == 2) {
      ss[2].visible = ss[5].visible = false;
    } else if (n == 3) {
      ss[4].visible = ss[5].visible = false;
    } else if (n == 4) {
      ss[0].visible = ss[3].visible = ss[4].visible = false;
    } else if (n == 5) {
      ss[1].visible = ss[4].visible = false;
    } else if (n == 6) {
      ss[1].visible = false;
    } else if (n == 7) {
      ss[3].visible = ss[4].visible = ss[5].visible =
ss[6].visible = false;
    } else if (n == 9) {
      ss[4].visible = false;
    }
  }

  this.add = function(n) {
    currentNo += n;
    currentNo = currentNo % 10;
    this.set(currentNo);
  };

  this.set(0);
}

function Score(origin) {
  var digit1 = new SSD(origin);
  var digit2 = new SSD(origin + new Point(100, 0));
  var value = 0;

  this.set = function(n) {
    digit1.set(Math.floor(n / 10));
    digit2.set(n % 10);
  };

  this.add = function(n) {
    value += n;

    if (value < 0) {
      value = 0;
    } else {
      value = value % 100;
```

```
    }

    this.set(value);
  };
}

var score = new Score(new Point(10, 10));

var addButton = new Game.Button('+', 0, 170);
var minusButton = new Game.Button('-', 50, 170);
minusButton.bounds.width = addButton.bounds.width;

addButton.onMouseDown = function() {
  score.add(1);
};

minusButton.onMouseDown = function() {
  score.add(-1);
};
```