Graphics and Games

#WWDC14

Best Practices for Building SpriteKit Games

Session 608

Jacques Gasselin de Richebourg
Game Technologies Manager

Nick Porcino
Senior Game Technologies Engineer

Scalability Best Practices

Game Structure Best Practices

Performance Best Practices

Scalability Best Practices

Problem—Hardcoding

Problem—Hardcoding

One scene does all the work

All art referenced in code

Level 1 was hardcoded

Level 2 was a lot of work

Level 3 looks the same

Stuck on Level 4

Problem—Coding Data

Problem—Coding Data

```
- (void)loadLevel1 {
    //clear the scene
    [scene removeAllChildren];

    //set the hero initial state
    hero.position = …
    hero.zRotation = …
    [scene addChild:hero];

    //set the enemy initial state
    enemy[0].position = …
    enemy[1].position = …
    …

}
```

Why Is This Bad?

Why Is This Bad?

Changing art assets means changing code

Visual feedback only via Build and Run

Designers must be programmers

Down the Line

Down the Line

Duplicates structural code

Code as data is not efficient

Hard to change collaboratively

The Solution

The Solution

Separate game content from game logic

Separate scene structure from assets

Separate data from code

Visualize in Xcode

Implement the Solution

Implement the Solution

Game logic in MyScene.m

Game scene structure in MyScene.sks

Scene assets in separate .sks files

Keep data in .plist files

Tools We Provide in Xcode 6

Tools We Provide in Xcode 6

SpriteKit template

SpriteKit editor

- Visual feedback is immediate
- Live physics simulation

.plist data editor

- XML during development
- Binary on deploy

Demo

SpriteKit Template

Recap

SpriteKit Template

Recap

Keeps scene content and game logic separated

Makes a scene file to edit structure

`MyScene.h` `MyScene.m` **`MyScene.sks`**

Loads it as your scene class

```
NSData *sceneData = …
NSKeyedUnarchiver *arch = [[NSKeyedUnarchiver alloc]
      initForReadingWithData:sceneData];
[arch setClass:MyScene.class forClassName:@“SKScene”];
MyScene *scene = [arch decodeObjectForKey:NSKeyedArchiveRootObjectKey];
[arch finishDecoding];
```

Game Structure Best Practices

Motivation

Motivation

Get your game running on the first day

Without compromising on scalability

Set you up to iterate collaboratively

1. Make Your Generic Level
2. Add Placeholder Content
3. Hook up Interactions
4. Get the Game Logic Working
5. Finish the Game

Make Your Generic Level

Make Your Generic Level

Logical layout only

Place markers—empty nodes with logical names

- Where the hero begins
- Where enemies appear
- Logical layers in the scene

Add Placeholder Content

Add Placeholder Content

Add colored SpriteNodes, without a texture, where visual elements are

Keep colors consistent

- Heroes in blue
- Enemies in red

Make parent-child relationships

- Particle emission locations
- Attachment points
- Armatures—arms, legs

Hook Up Interactions

Hook Up Interactions

This pass ensures the physics interaction is right

Setup physics properties

- Categories
- Collision masks
- Static vs. Dynamic

Simulate physics in Xcode

Get the Game Logic Working

Get the Game Logic Working

Initialize your scene logic and game logic

Hook scene objects to game objects

- Name your nodes
- Search for nodes by name in code

Insert hooks for overriding placeholder content

GoodHammer 〉 My Mac 64–bit | GoodHammer | Build GoodHammer: **Succeeded** | Today at 3:31 PM

GoodHammer 〉 GoodHammer 〉 GameScene.sks 〉 SKScene 〉 SKNode: totem

Manual 〉 GoodHammer 〉 GoodHammer 〉 GameScene.m 〉 -setupLevel1

```
28  @end
29
30  @implementation GameScene {
31      int score;
32      NSMutableArray *totemTouched;
33  }
34
35  - (void)didMoveToView:(SKView *)view {
36      [self setupScene];
37      [self setupLevel1];
38  }
39
40  - (void)setupScene {
41      score=0;
42      totemTouched = [NSMutableArray array];
43      [self.physicsWorld setContactDelegate:self];
44  }
45
46  - (void)setupLevel1 {
47      SKNode *totemNode = [self childNodeWithName:@"totem"];
48      SKScene *totemScene = [SKScene unarchiveFromFile:@"Level1_totem"];
49
50      [totemScene enumerateChildNodesWithName:@"totem" usingBlock:^(SKNode *node, BOOL *stop) {
51          [totemNode addChild:[node copy]];
52      }];
53  }
54
55  - (void)didBeginContact:(SKPhysicsContact *)contact {
56      if(contact.bodyA.node == [self childNodeWithName:@"hammer"] && [contact.bodyB.node.name
            isEqualToString:@"totem"]) {
57          if(![totemTouched containsObject:contact.bodyB.node]) {
58              score+=1;
59              [totemTouched addObject:contact.bodyB.node];
60          }
61      }
62      if(contact.bodyB.node == [self childNodeWithName:@"hammer"] && [contact.bodyA.node.name
            isEqualToString:@"totem"]) {
63          if(![totemTouched containsObject:contact.bodyA.node]) {
64              score+=1;
65              [totemTouched addObject:contact.bodyA.node];
66          }
67      }
68      SKLabelNode *label = (SKLabelNode*)[self childNodeWithName:@"scoreLabel"];
69      label.text = [NSString stringWithFormat:@"Score : %d",score];
70  }
71
72  -(void)mouseDown:(NSEvent *)theEvent {
73      /* Called when a mouse click occurs */
74      self.physicsWorld.speed = 0.5;
75
76      SKNode *hammer = [self childNodeWithName:@"hammer"];
77      hammer.physicsBody.affectedByGravity = true;
78      hammer.physicsBody.contactTestBitMask = 1;
79      [hammer.physicsBody applyImpulse:CGVectorMake(hammer.physicsBody.mass * 820,
80                                                    hammer.physicsBody.mass * 320)];
81      [hammer.physicsBody applyAngularImpulse:hammer.physicsBody.mass * -4.2];
82  }
83
```

Get the Game Logic Working

Details—Where to do this

Two logical points of initialization

- On first load `-initWithCoder:`
- On first shown `-didMoveToView:`

Get the Game Logic Working

Details—On first load

Get the Game Logic Working

Details—On first load

Called automatically by NSKeyedUnarchiver when the scene is loaded

Load any side-car data like sounds or AI data here

```
- (instancetype)initWithCoder:(NSCoder *)aCoder {
    self = [super init];
    if (self) {
        NSArray *enemyStats = [NSArray arrayWithContentsOfFile:
                                                <path to stats plist>];
    }
    return self;
}
```

Get the Game Logic Working

Details—On first shown

Get the Game Logic Working

Details—On first shown

Called when SKView.presentScene: is called

Cache visual elements

```
- (void)didMoveToView:(SKView *)view {
    self.enemies = [NSMutableArray new];
    [self enumerateChildNodesWithName:@“//enemy*”
                           usingBlock:^(SKNode *node, BOOL *stop) {
       [self.enemies addObject:node];
    }];
}
```

Enumerating Logical Elements

Details—Motivation

Enumerating Logical Elements

Details—Motivation

Hooks logical scene elements to your code using search

- `-childNodeNamed` for a single element
- `-enumerateChildNodesWithName` for multiple elements

Cache results

Enumerating Logical Elements

Details—Quick intro on search syntax

Enumerating Logical Elements

Details—Quick intro on search syntax

Search by name

- `@“hero”` finds a child called hero, without recursion
- `@“//hero”` finds all nodes in the scene graph called hero, recursively

Search by class

- `@“//SKEmitterNode”` would find all emitters in the scene graph

Search with wildcard

- `@“//he*”` would find all nodes beginning with he

GoodHammer 〉 My Mac 64–bit | GoodHammer | Build GoodHammer: **Succeeded** | Today at 3:31 PM

GoodHammer 〉 GoodHammer 〉 GameScene.sks 〉 SKScene 〉 SKNode: totem

Manual 〉 GoodHammer 〉 GoodHammer 〉 GameScene.m 〉 -setupLevel1

```
28  @end
29
30  @implementation GameScene {
31      int score;
32      NSMutableArray *totemTouched;
33  }
34
35  - (void)didMoveToView:(SKView *)view {
36      [self setupScene];
37      [self setupLevel1];
38  }
39
40  - (void)setupScene {
41      score=0;
42      totemTouched = [NSMutableArray array];
43      [self.physicsWorld setContactDelegate:self];
44  }
45
46  - (void)setupLevel1 {
47      SKNode *totemNode = [self childNodeWithName:@"totem"];
48      SKScene *totemScene = [SKScene unarchiveFromFile:@"Level1_totem"];
49
50      [totemScene enumerateChildNodesWithName:@"totem" usingBlock:^(SKNode *node, BOOL *stop) {
51          [totemNode addChild:[node copy]];
52      }];
53  }
54
55  - (void)didBeginContact:(SKPhysicsContact *)contact {
56      if(contact.bodyA.node == [self childNodeWithName:@"hammer"] && [contact.bodyB.node.name
            isEqualToString:@"totem"]) {
57          if(![totemTouched containsObject:contact.bodyB.node]) {
58              score+=1;
59              [totemTouched addObject:contact.bodyB.node];
60          }
61      }
62      if(contact.bodyB.node == [self childNodeWithName:@"hammer"] && [contact.bodyA.node.name
            isEqualToString:@"totem"]) {
63          if(![totemTouched containsObject:contact.bodyA.node]) {
64              score+=1;
65              [totemTouched addObject:contact.bodyA.node];
66          }
67      }
68      SKLabelNode *label = (SKLabelNode*)[self childNodeWithName:@"scoreLabel"];
69      label.text = [NSString stringWithFormat:@"Score : %d",score];
70  }
71
72  -(void)mouseDown:(NSEvent *)theEvent {
73      /* Called when a mouse click occurs */
74      self.physicsWorld.speed = 0.5;
75
76      SKNode *hammer = [self childNodeWithName:@"hammer"];
77      hammer.physicsBody.affectedByGravity = true;
78      hammer.physicsBody.contactTestBitMask = 1;
79      [hammer.physicsBody applyImpulse:CGVectorMake(hammer.physicsBody.mass * 820,
80                                                    hammer.physicsBody.mass * 320)];
81      [hammer.physicsBody applyAngularImpulse:hammer.physicsBody.mass * -4.2];
82  }
83
```

Finish the Game

Finish the Game

Add artwork

- Set the textures on your “red boxes”

Add levels

Add effects

- Shaders as .fsh files
- CoreImage filters

Play test—iterate

Have fun!

Performance Best Practices

Nick Porcino
Senior Game Technologies Engineer

Performance Best Practices

Performance Best Practices

Drawing performance

Actions and constraints

Physics

Shapes

Effects

Lighting

Drawing Performance

Drawing Performance

Drawing Performance

Two factors dominate

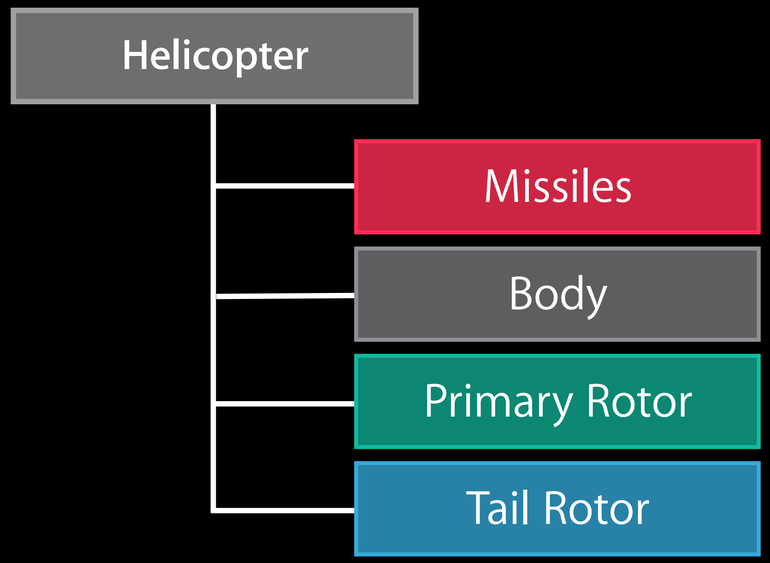
- Draw order
- Sharing

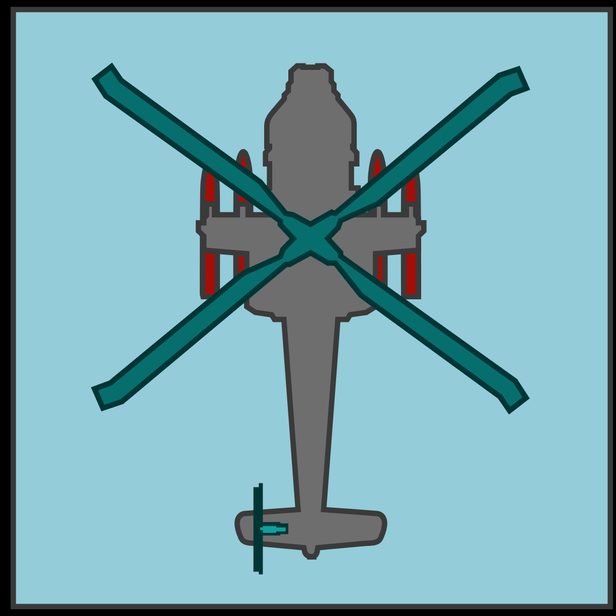
Draw Order

```
view.ignoresSiblingOrder = NO;
```

Two simple rules

- A parent draws its content before rendering its children
- Children are rendered in the order they appear in the child array





Drawing

Using sibling order

Drawing

Using sibling order

Drawing

Using sibling order

Drawing

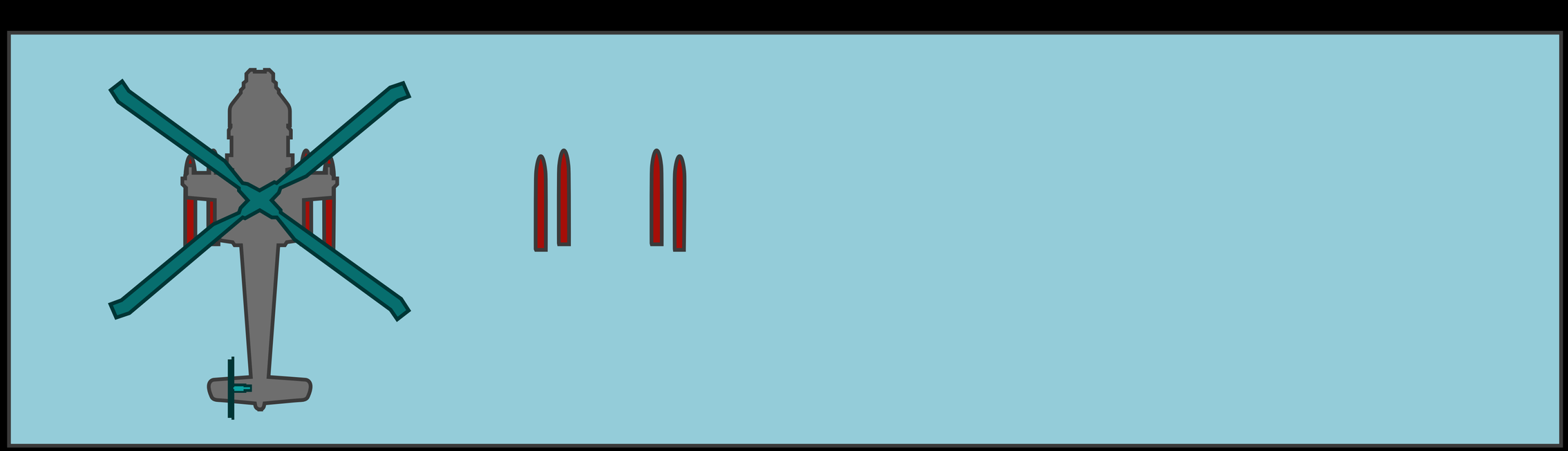
Using sibling order

Drawing

Using sibling order

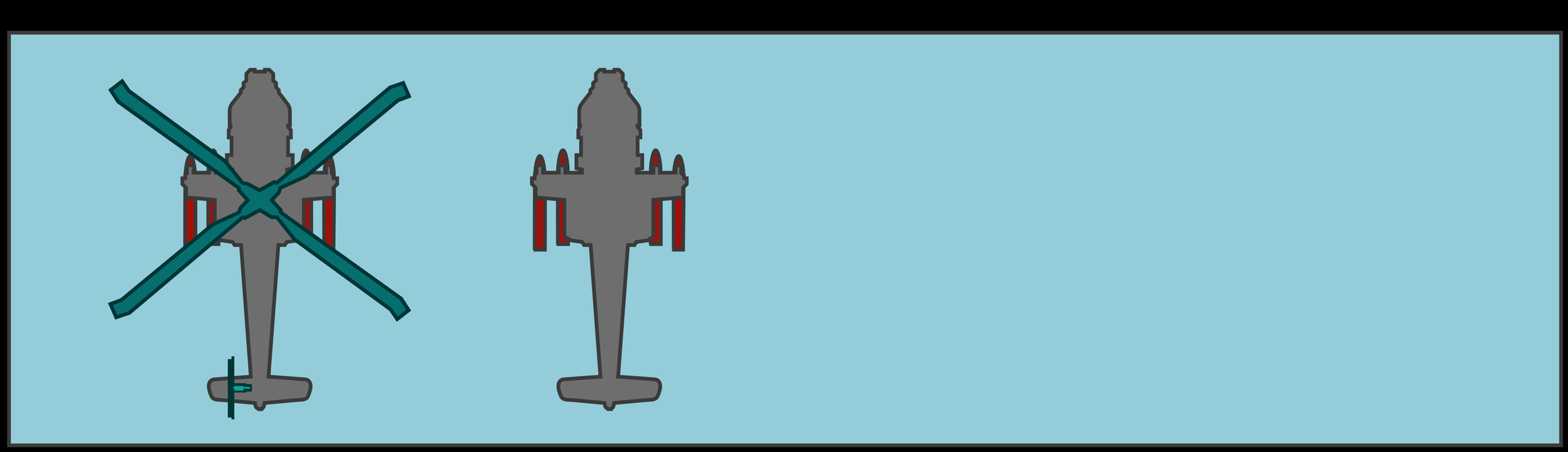
Drawing

Using sibling order



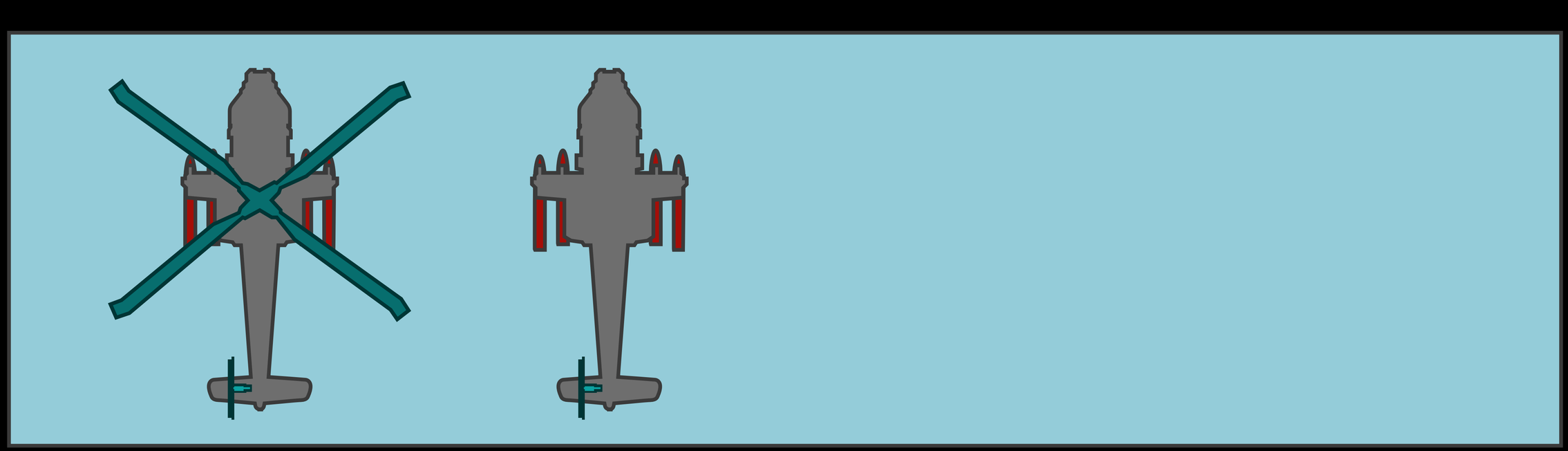
Drawing

Using sibling order



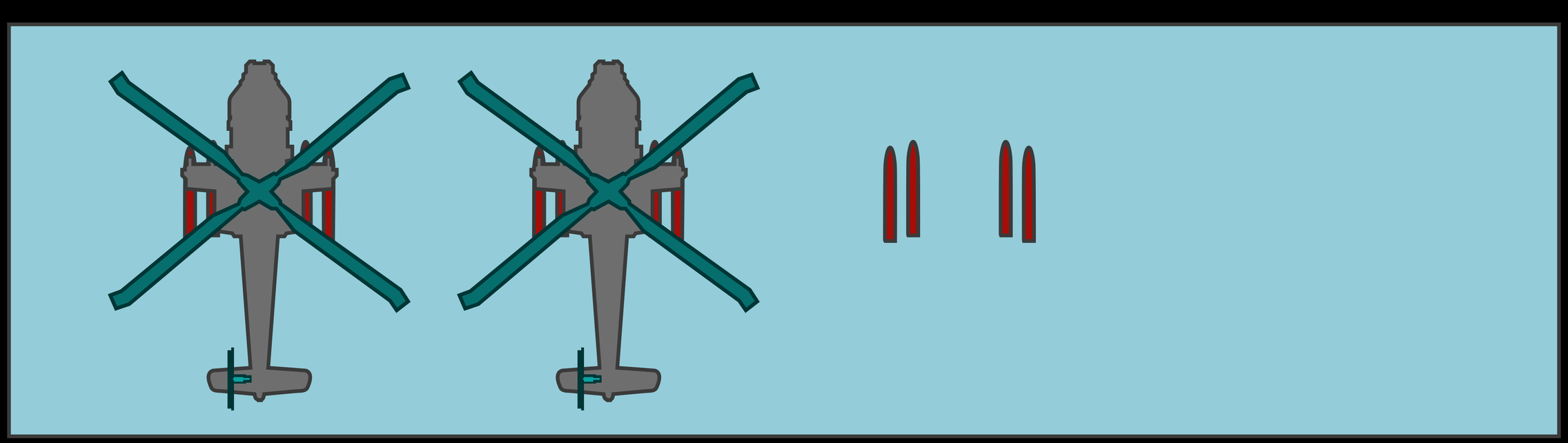
Drawing

Using sibling order



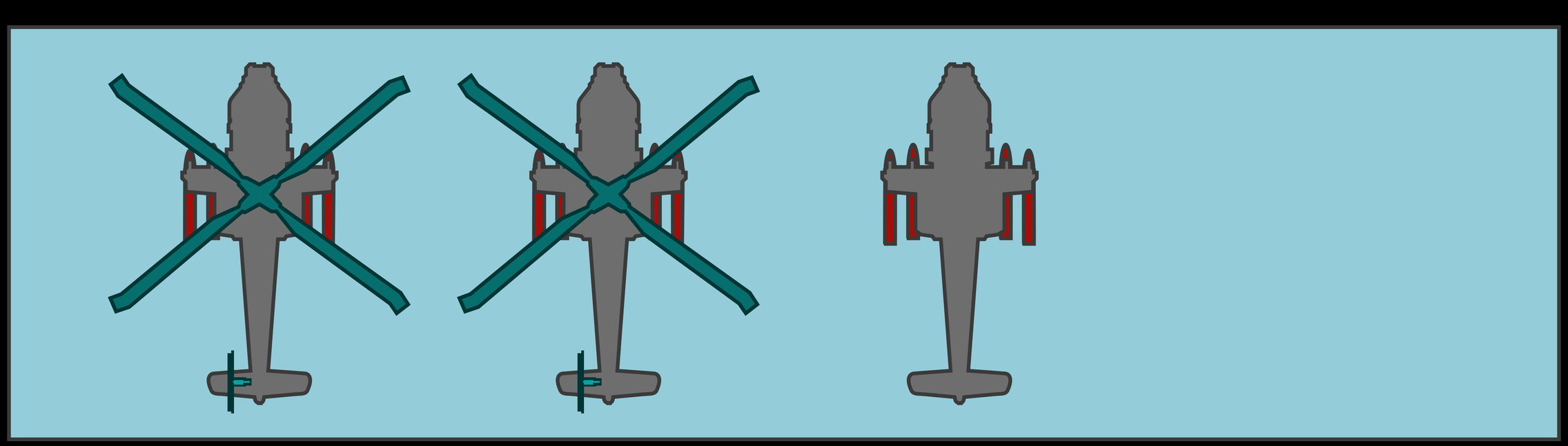
Drawing

Using sibling order



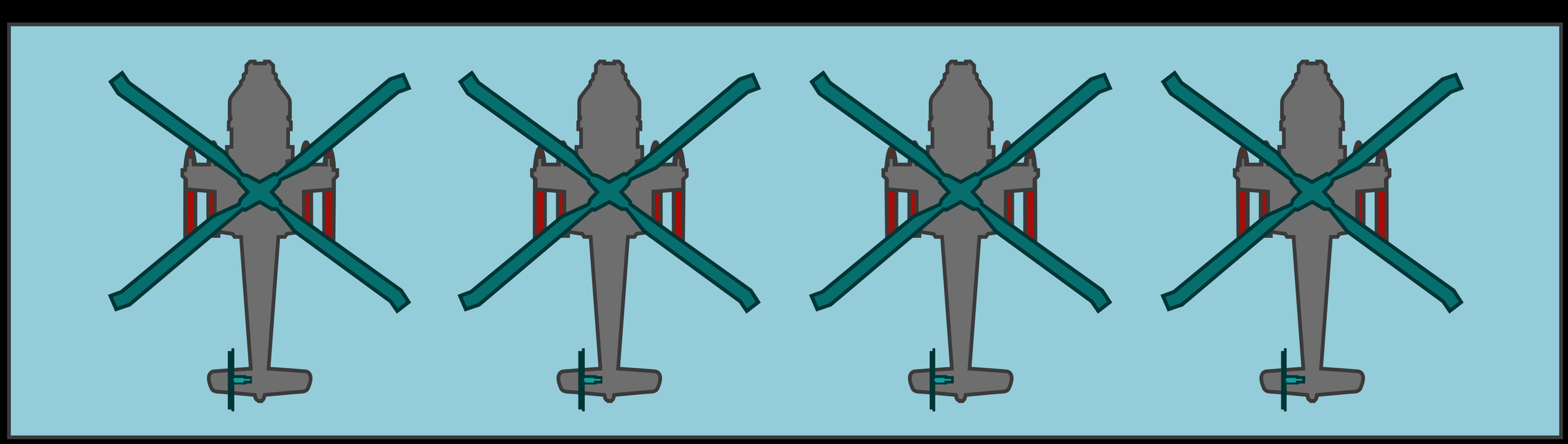
Drawing

Using sibling order



Drawing

Using sibling order



Draw Order

SpriteKit games are 2D

X, Y

Draw Order

Introducing…

X,Y

Draw Order

Introducing… The Third Dimension!

X, Y, Z

Draw Order

Using depth order

Draw Order

Using depth order

Activate depth order drawing with

```
view.ignoresSiblingOrder = YES;
```

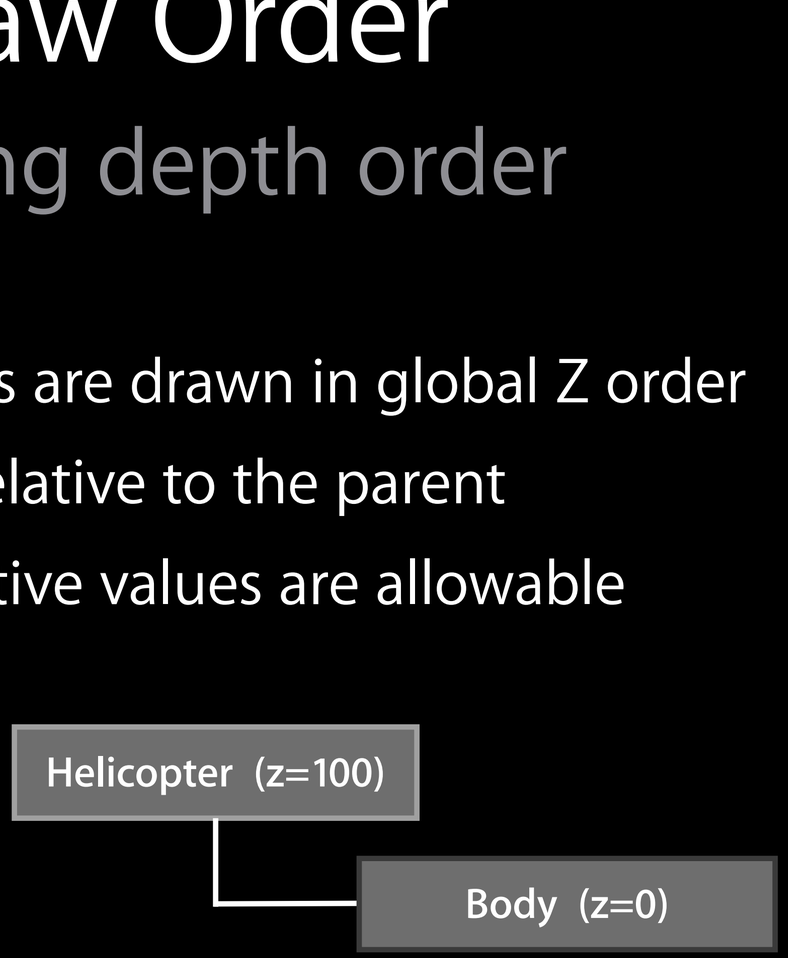
Draw Order

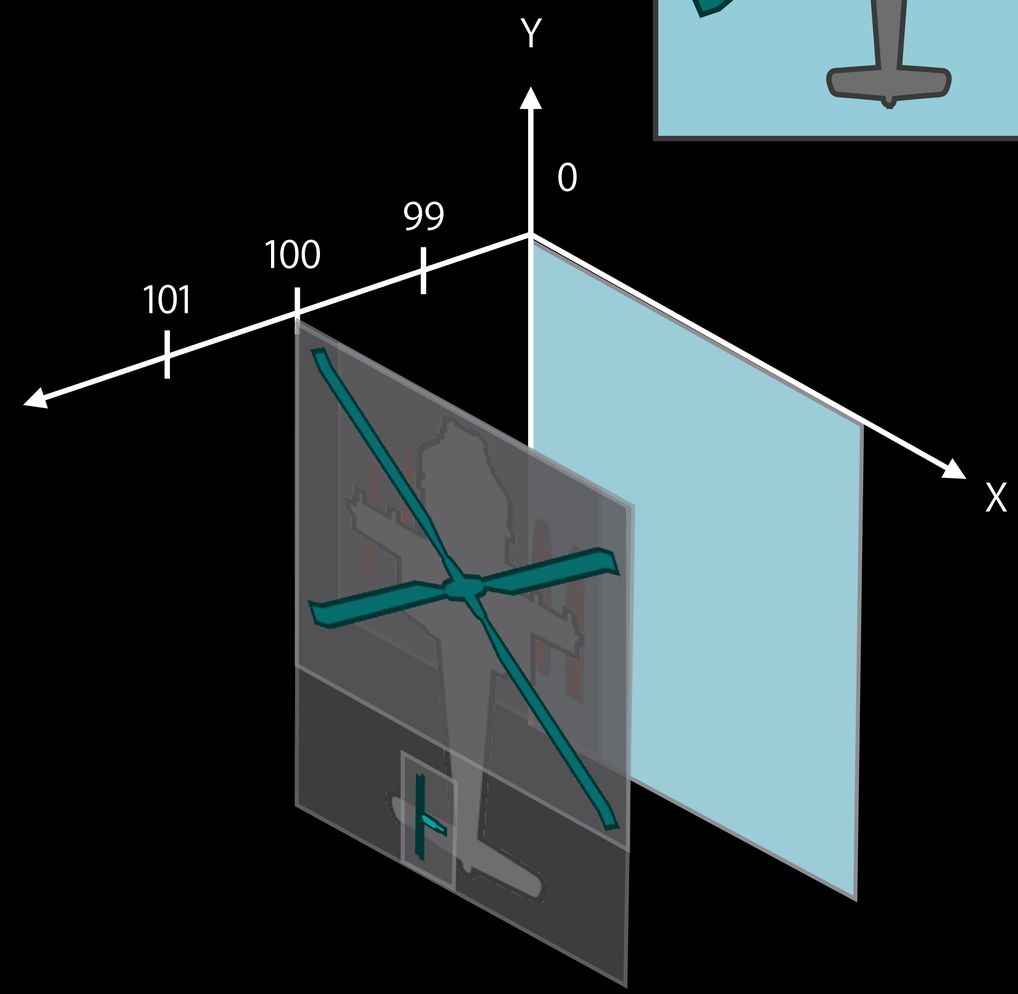
Using depth order

Nodes are drawn in global Z order

Z is relative to the parent

Negative values are allowable





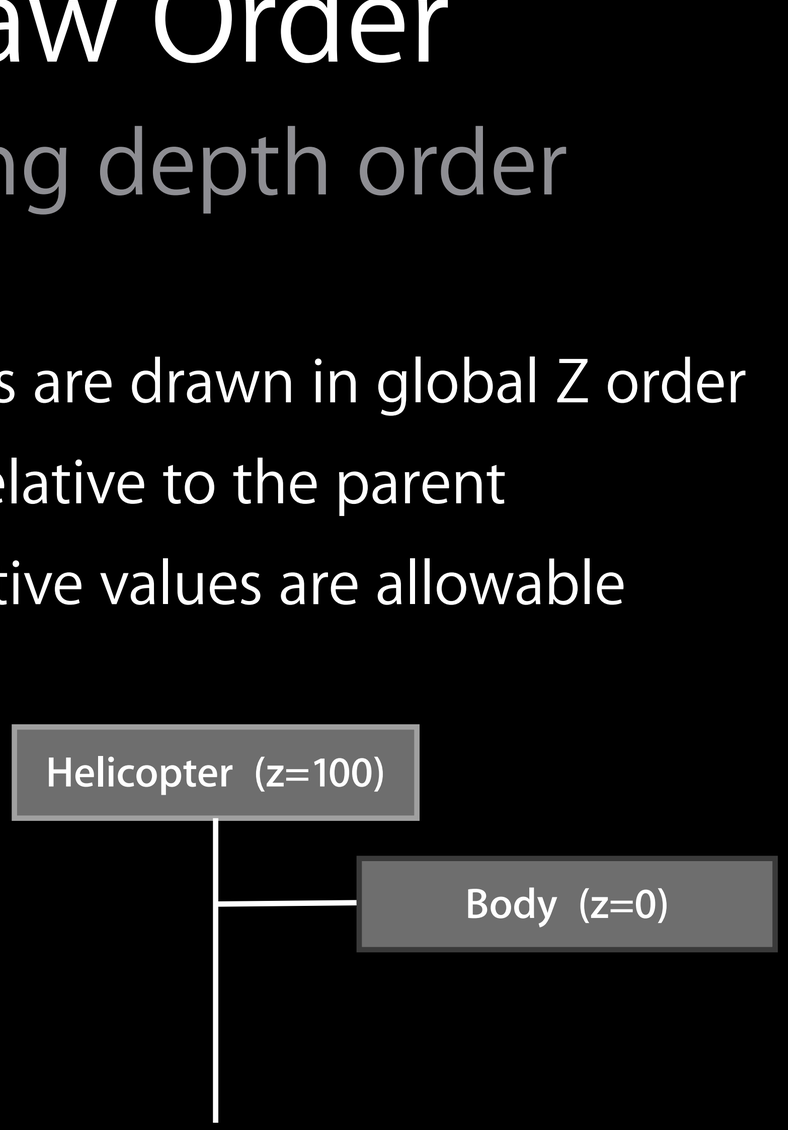
Draw Order

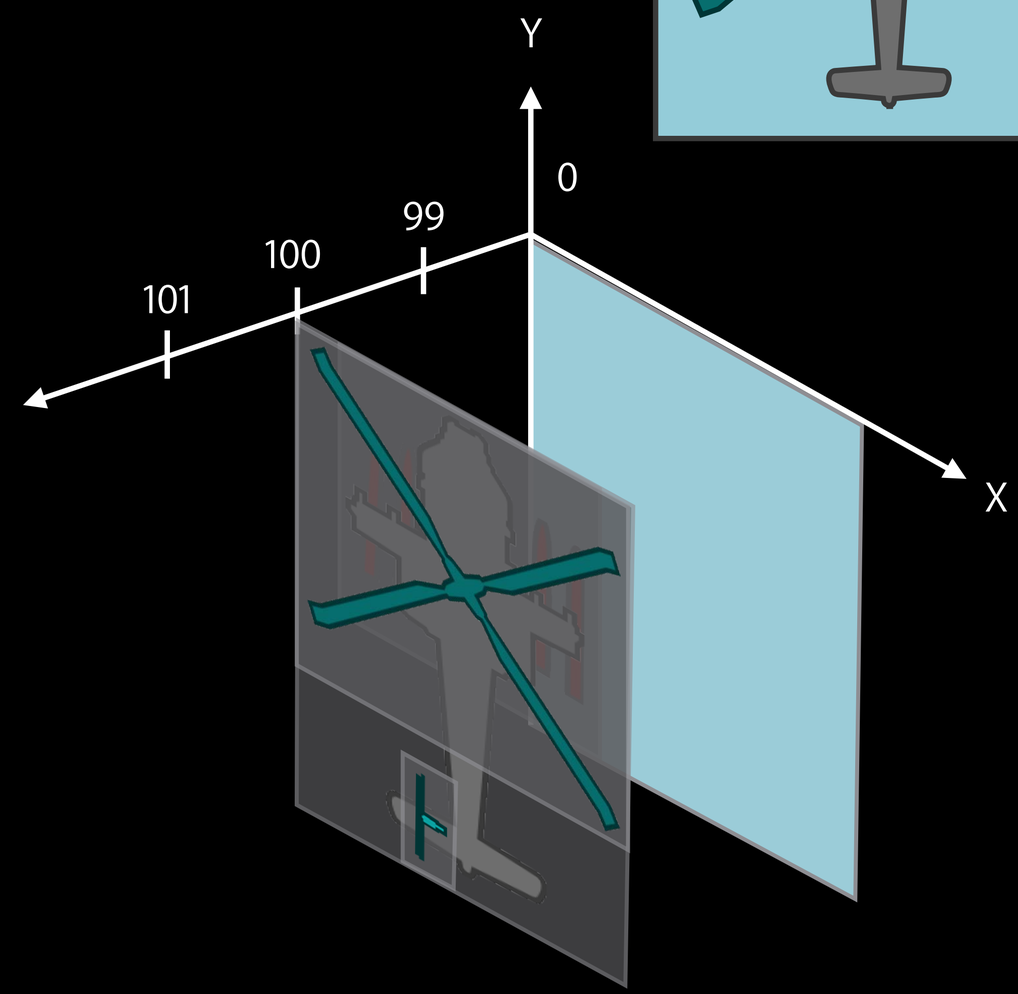
Using depth order

Nodes are drawn in global Z order

Z is relative to the parent

Negative values are allowable





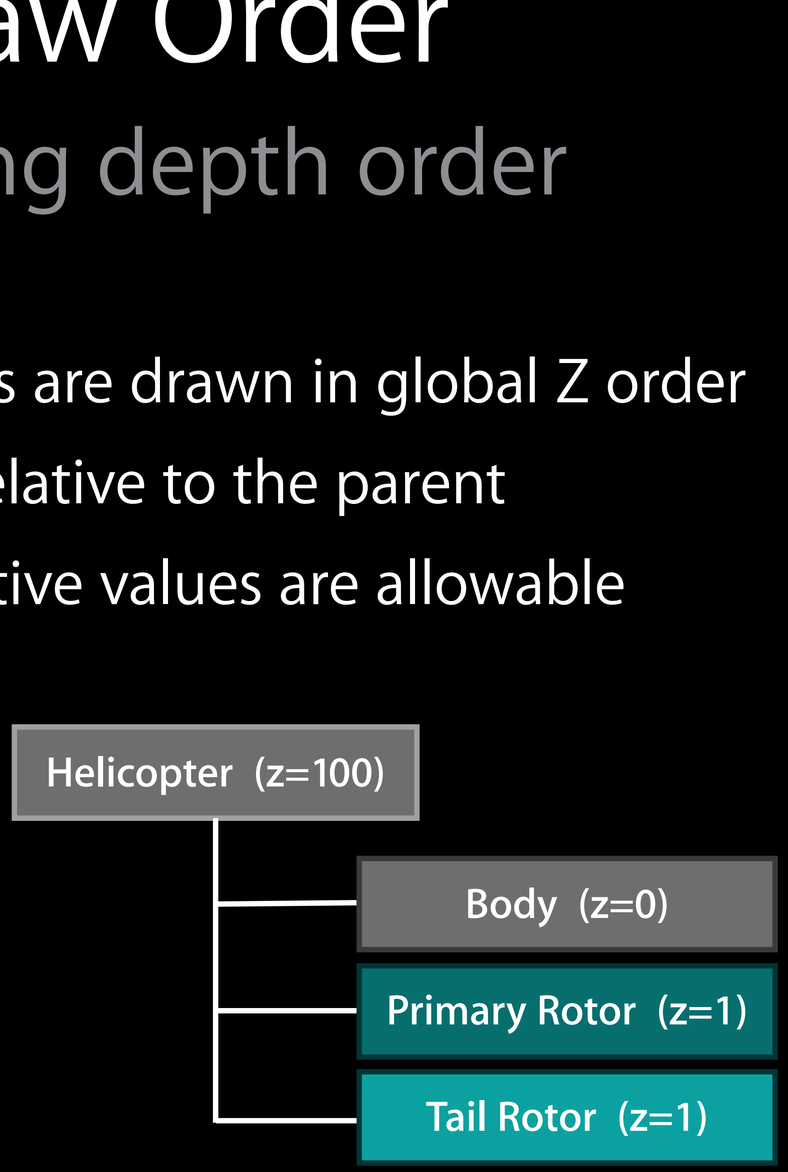
Draw Order

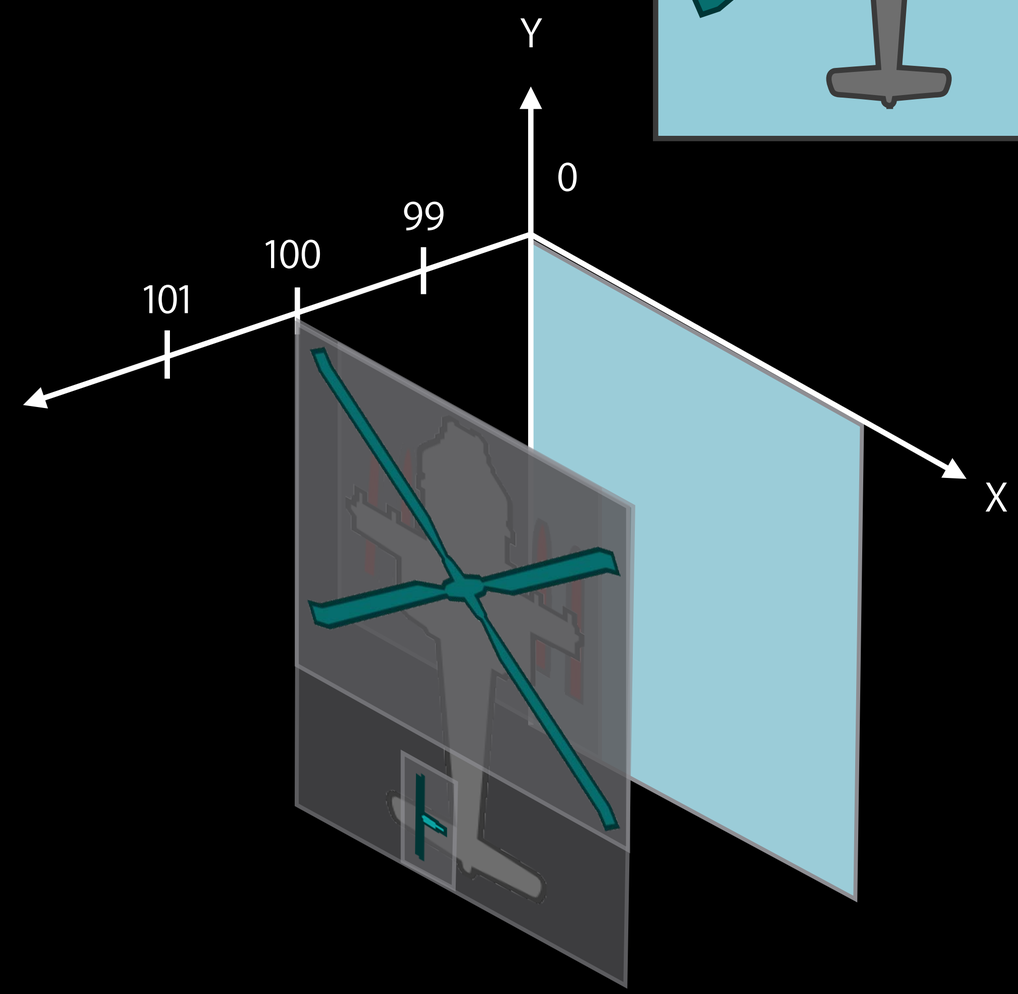
Using depth order

Nodes are drawn in global Z order

Z is relative to the parent

Negative values are allowable





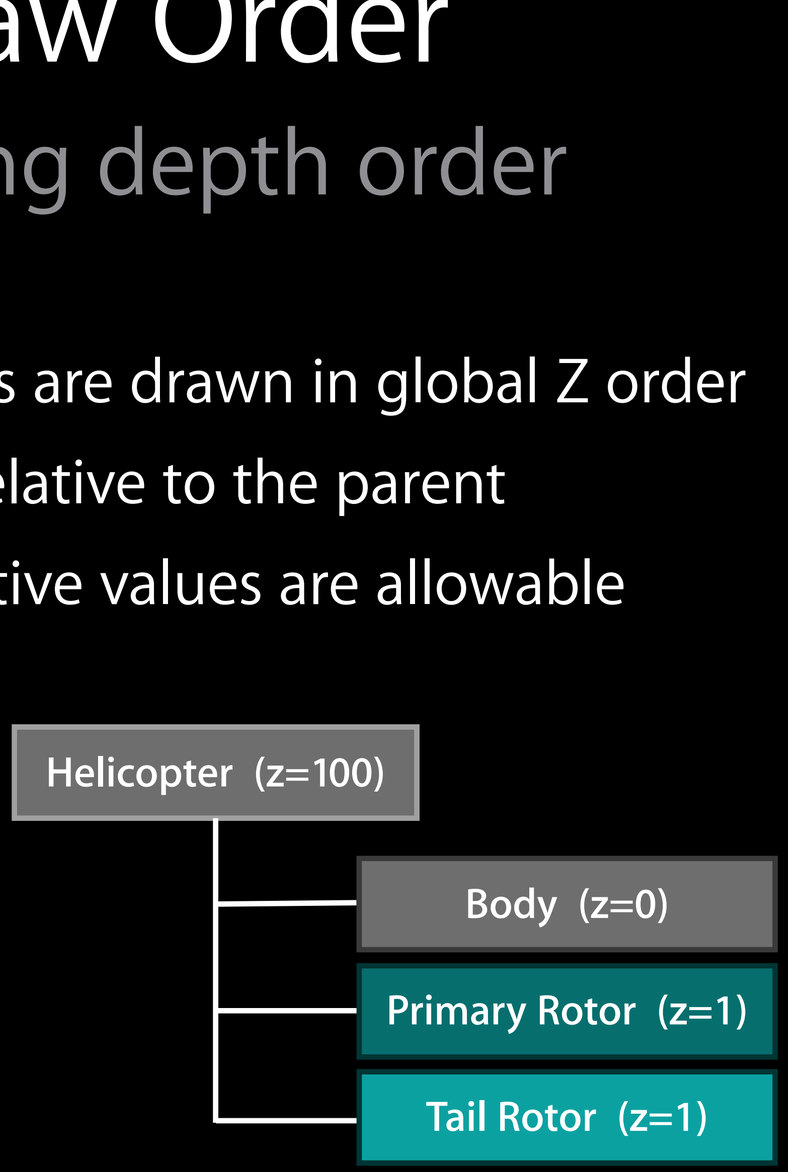
Draw Order

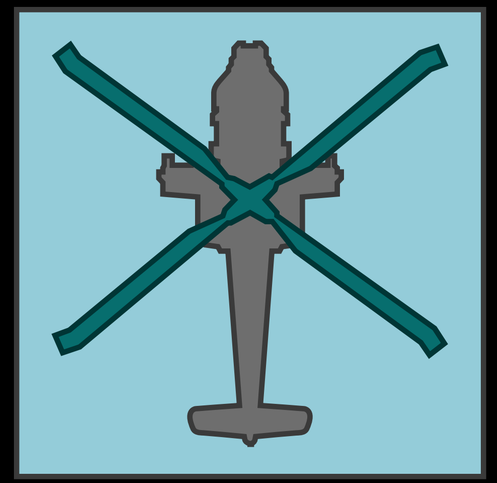
Using depth order

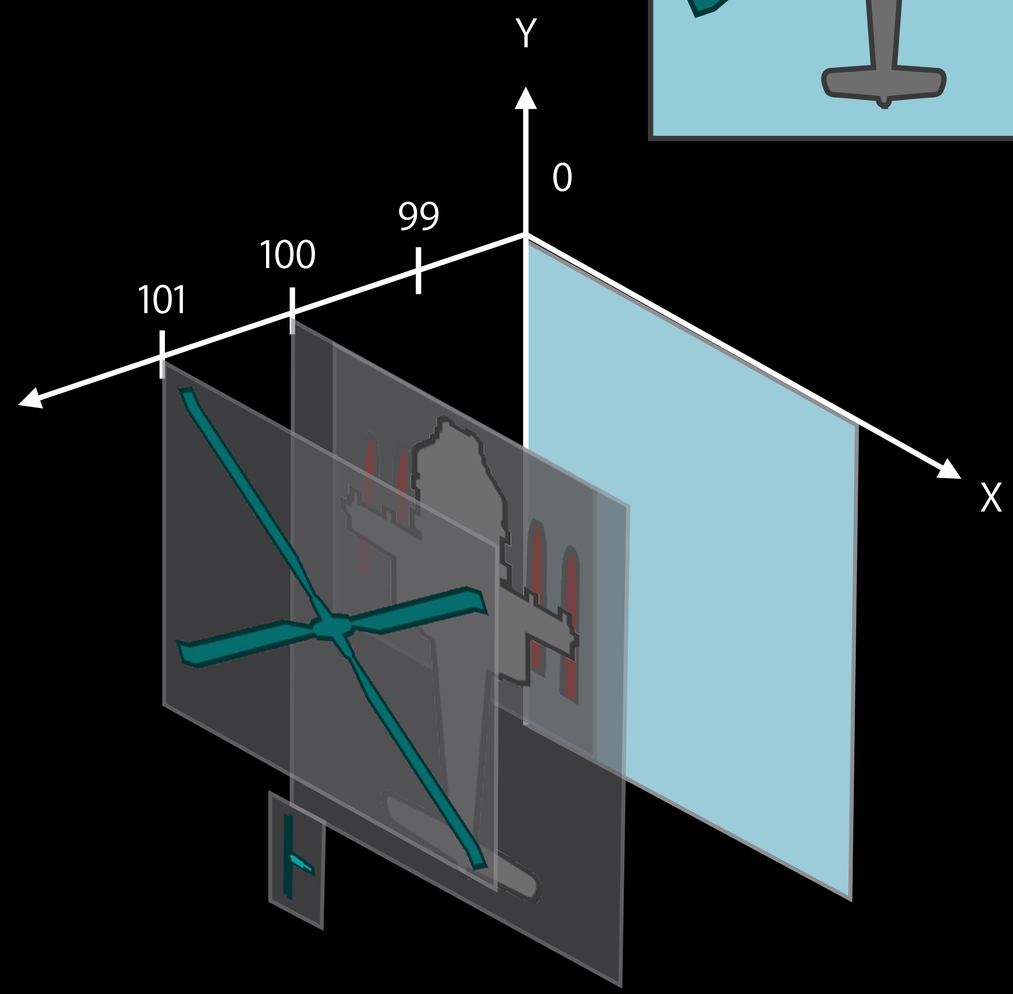
Nodes are drawn in global Z order

Z is relative to the parent

Negative values are allowable







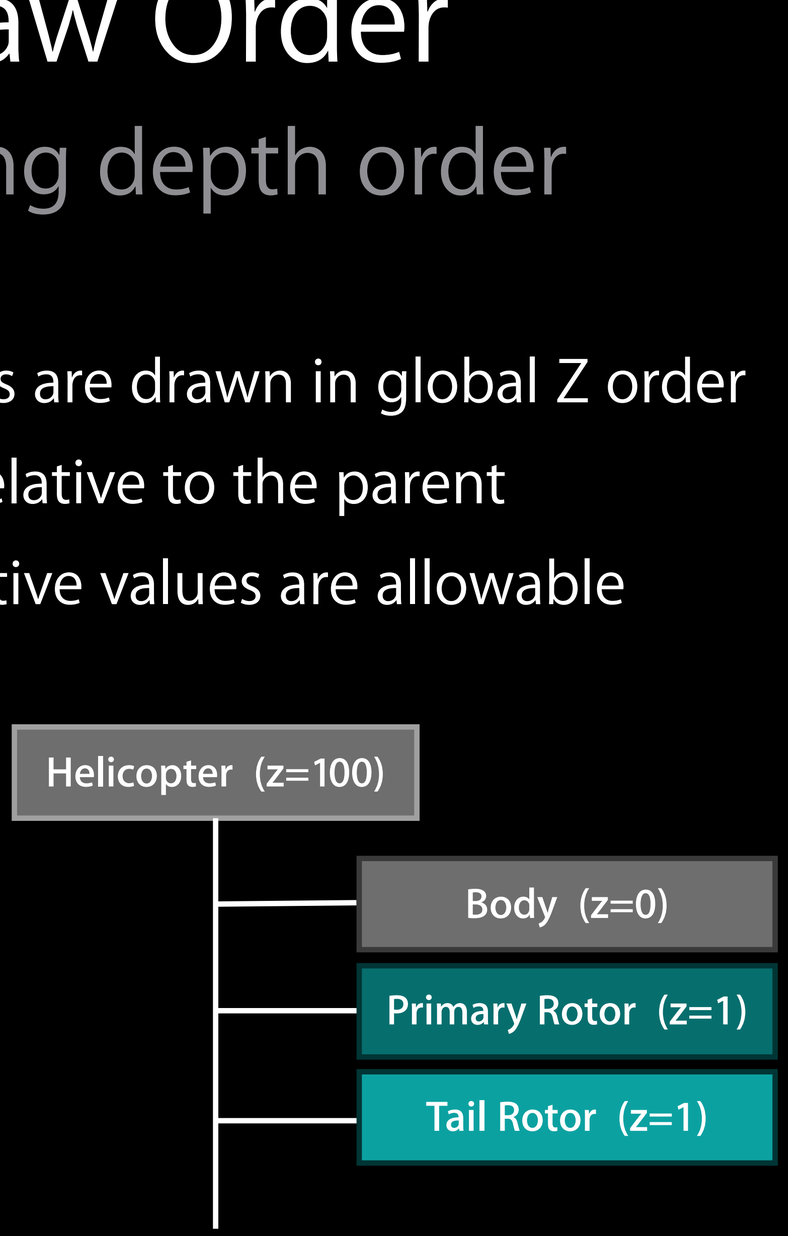
Draw Order

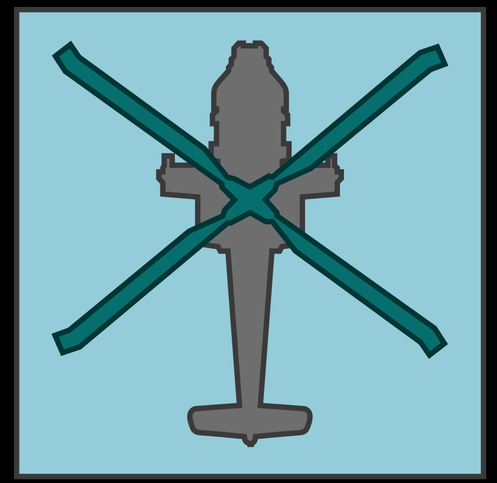
Using depth order

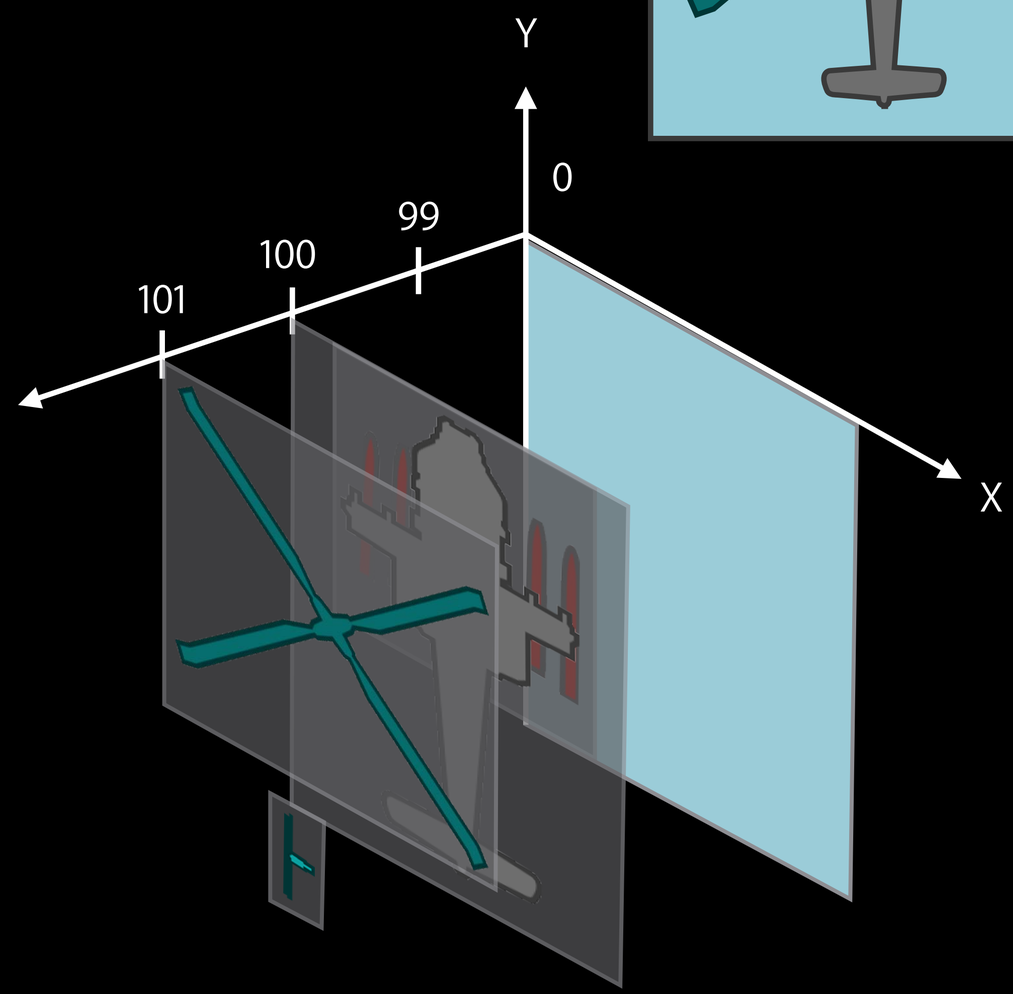
Nodes are drawn in global Z order

Z is relative to the parent

Negative values are allowable







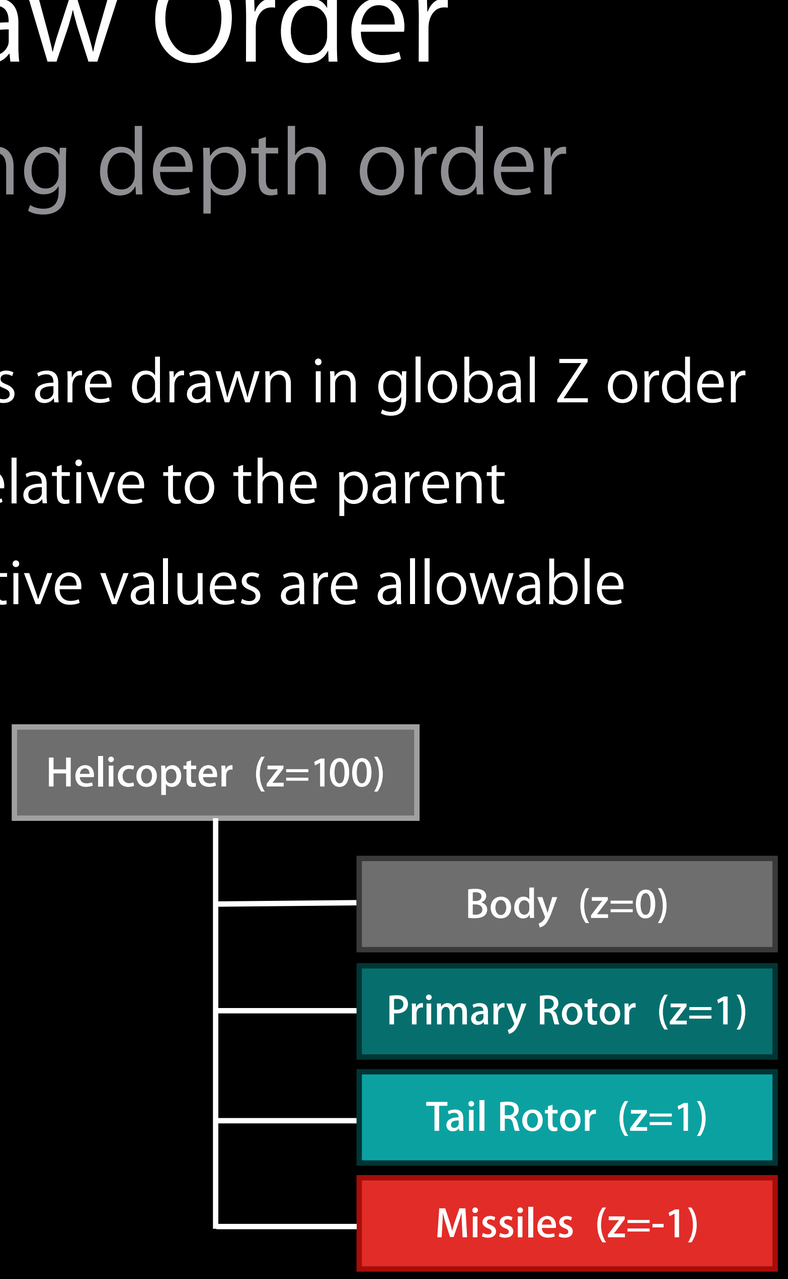
Draw Order

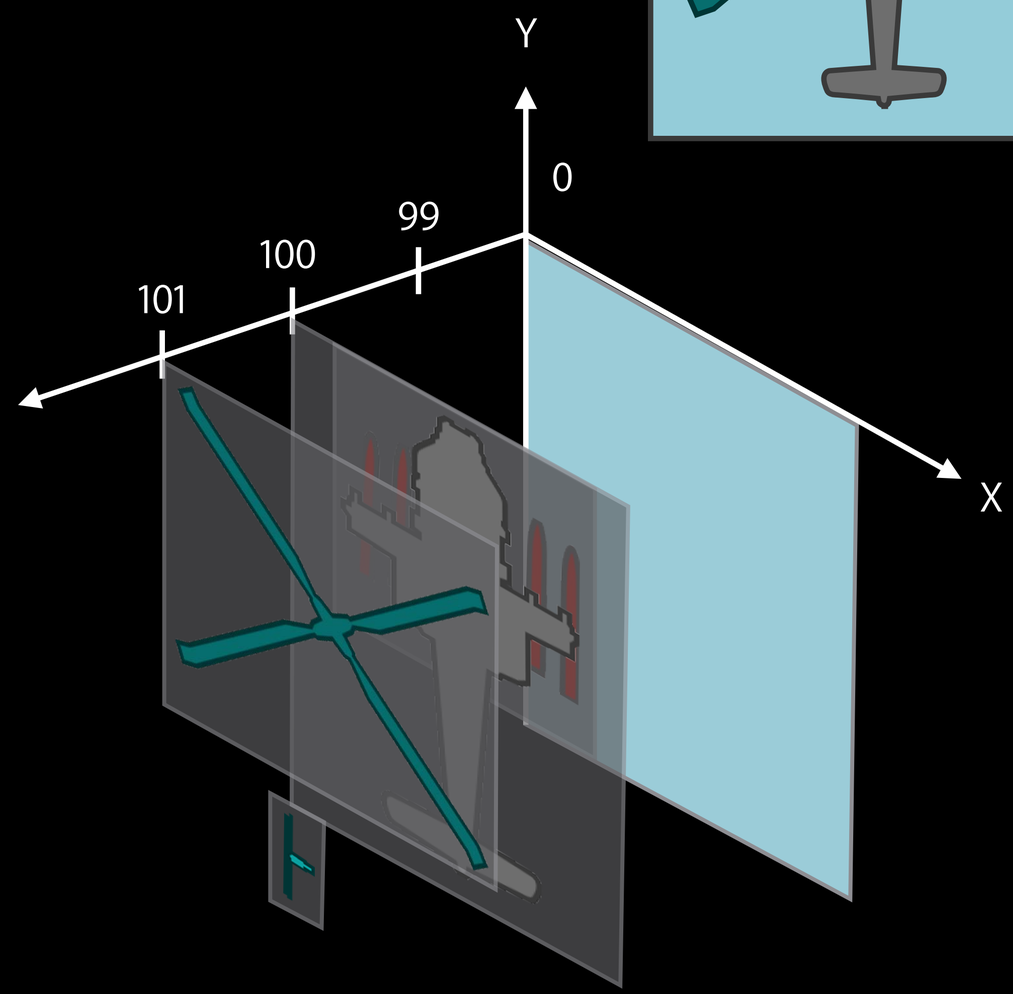
Using depth order

Nodes are drawn in global Z order

Z is relative to the parent

Negative values are allowable





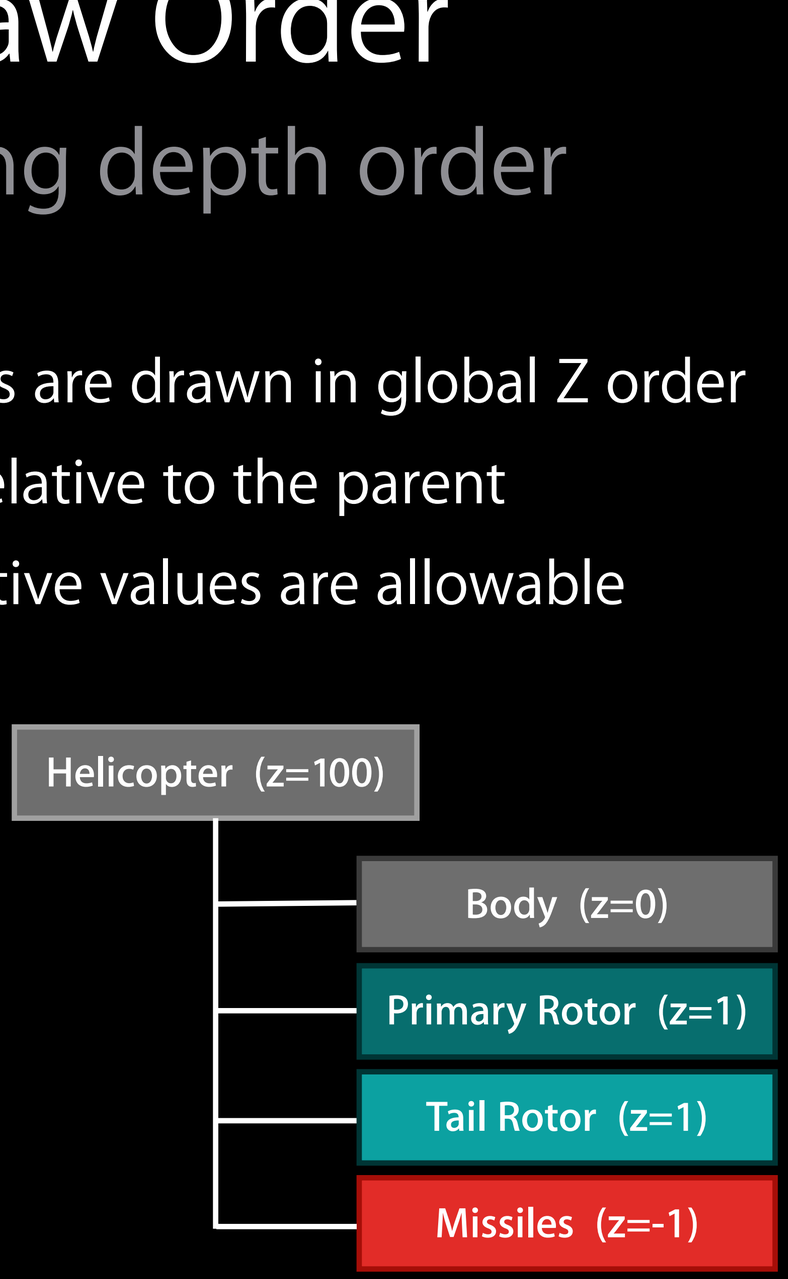
Draw Order

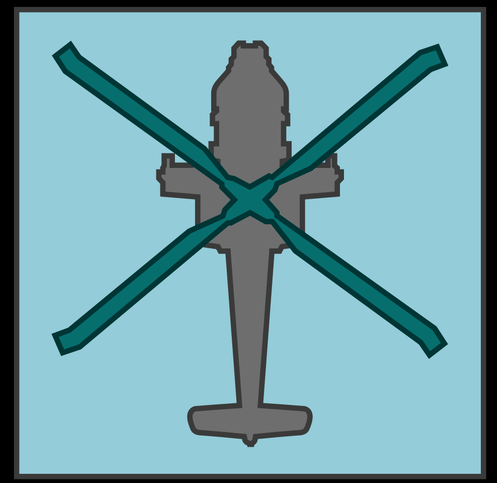
Using depth order

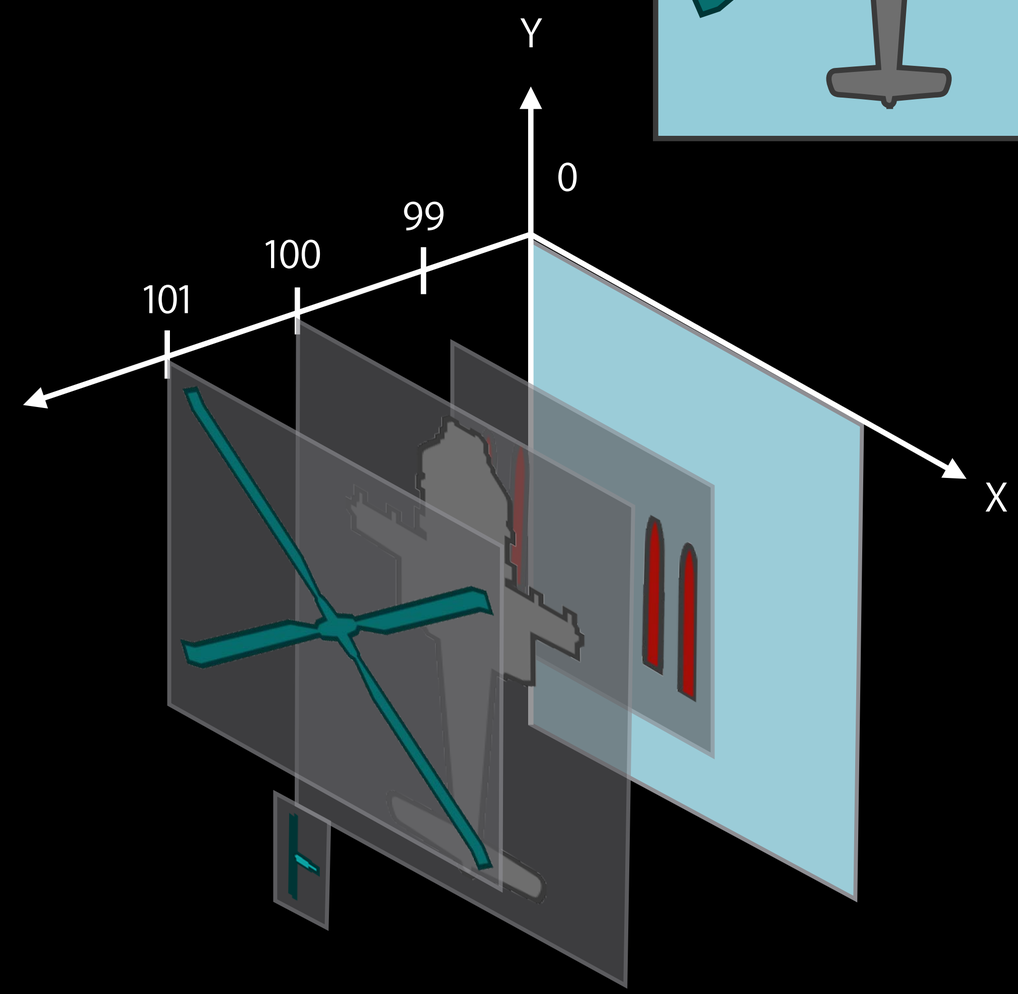
Nodes are drawn in global Z order

Z is relative to the parent

Negative values are allowable





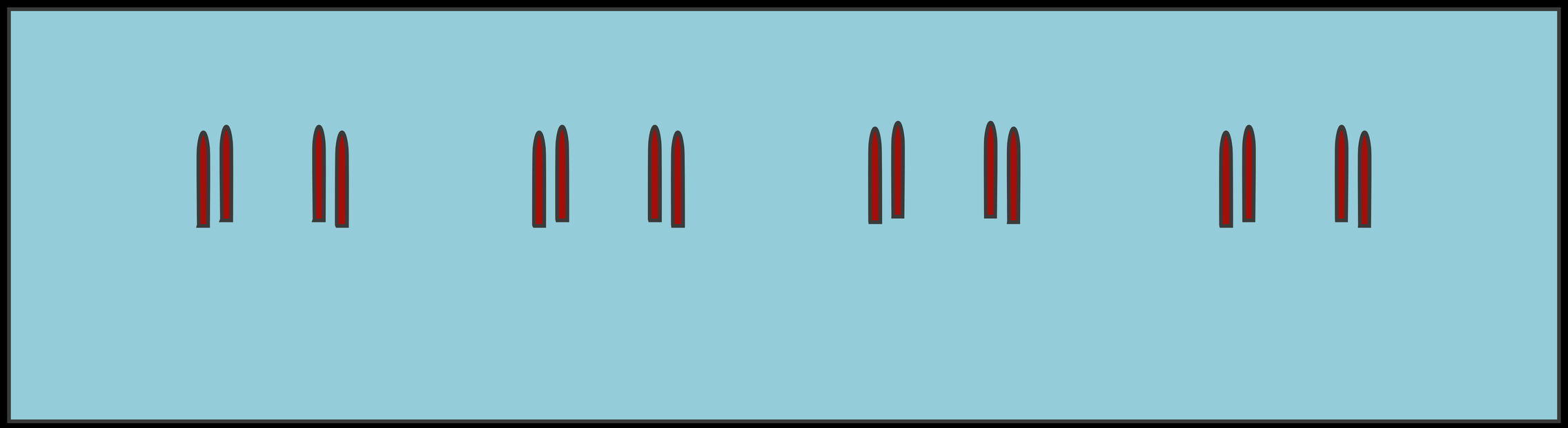


Batching

Using depth order

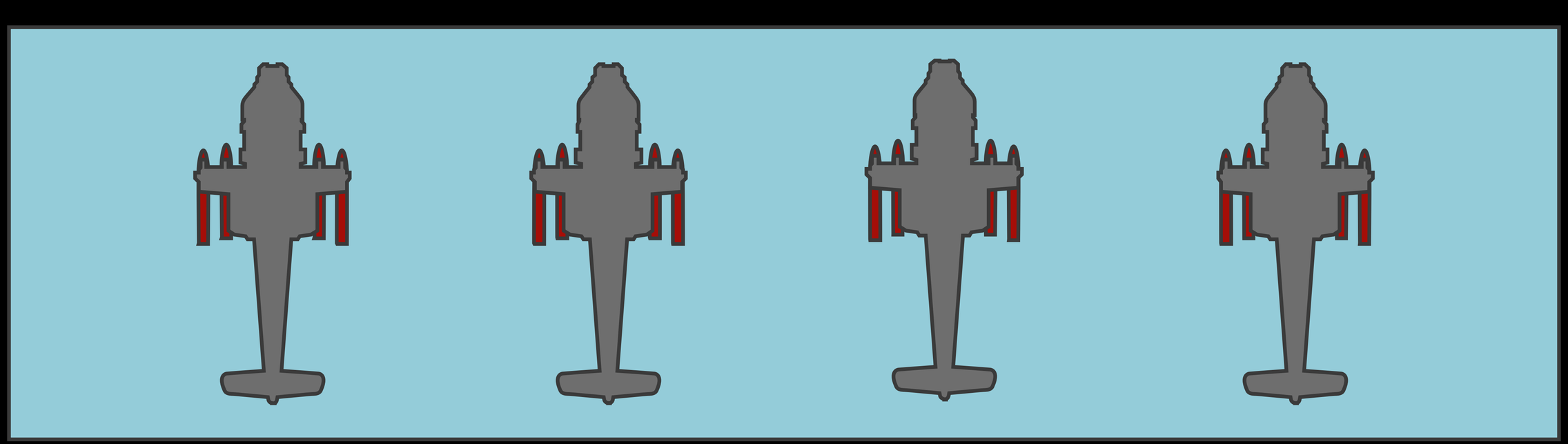
Batching

Using depth order



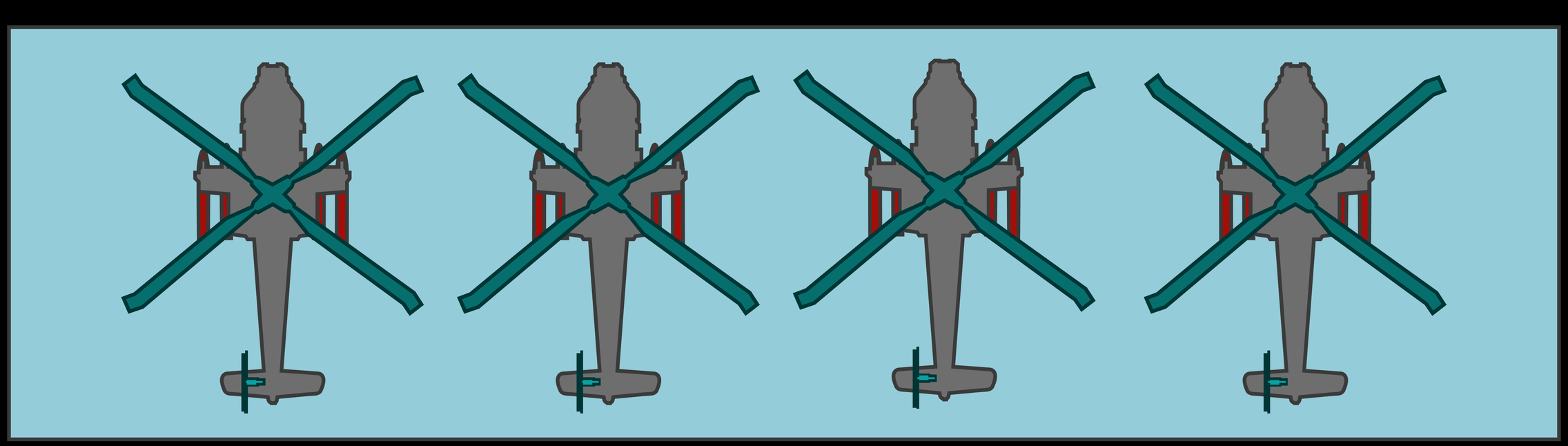
Batching

Using depth order



Batching

Using depth order

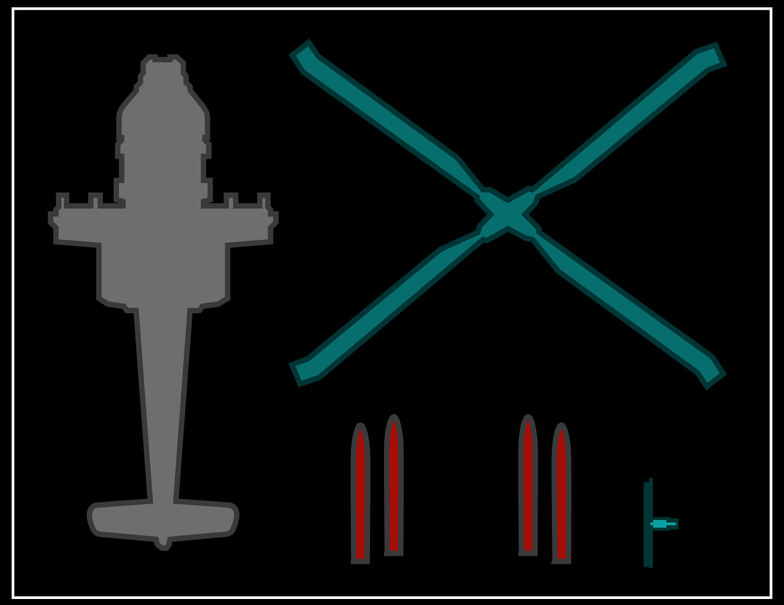


Optimize Batching

Texture sharing

Use the same texture on multiple sprites

Put textures into atlases

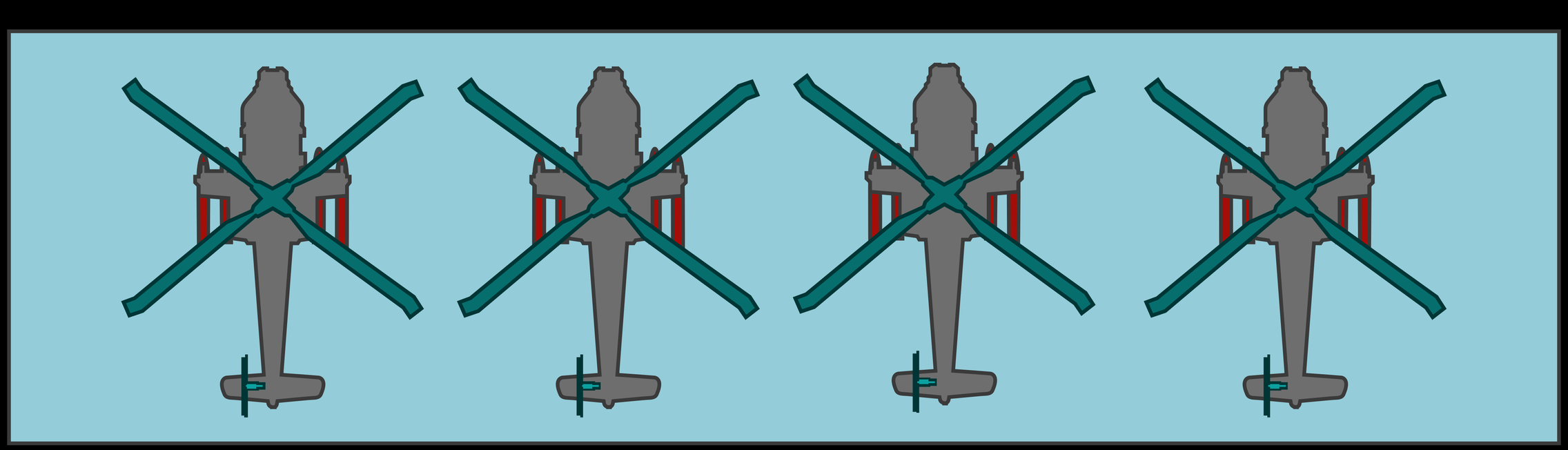


Batching

Draw order with sharing

Batching

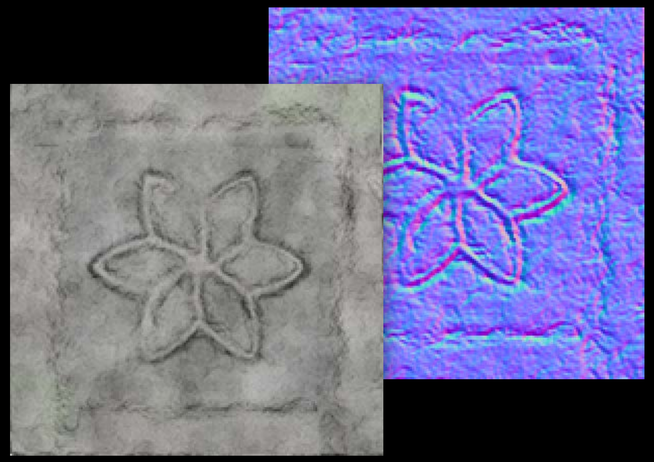
Draw order with sharing

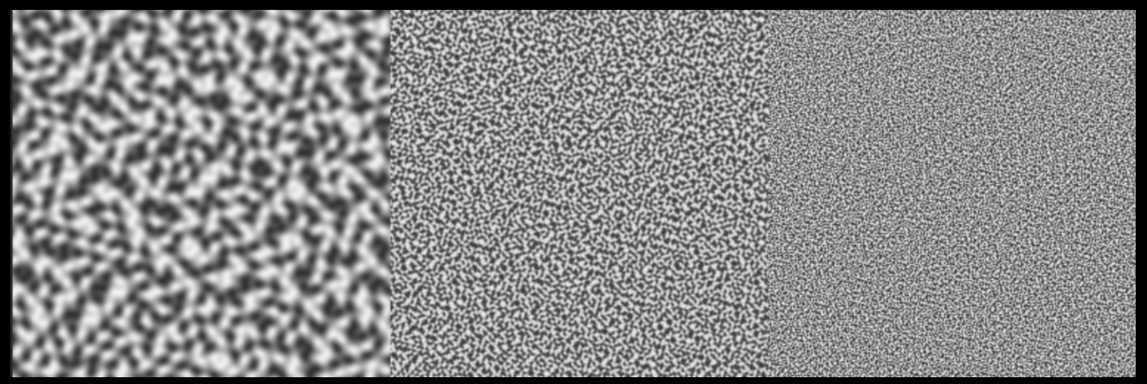


Optimize Batching

Optimize Batching

Keep a reference to generated normal maps and procedural noise





Optimize Batching

Optimize Batching

Reference shaders from files, not strings

Put blend modes such as `SKBlendModeAdd` on the same depth layer

Tools

Tools

Tools to help evaluate graphics performance

Tools

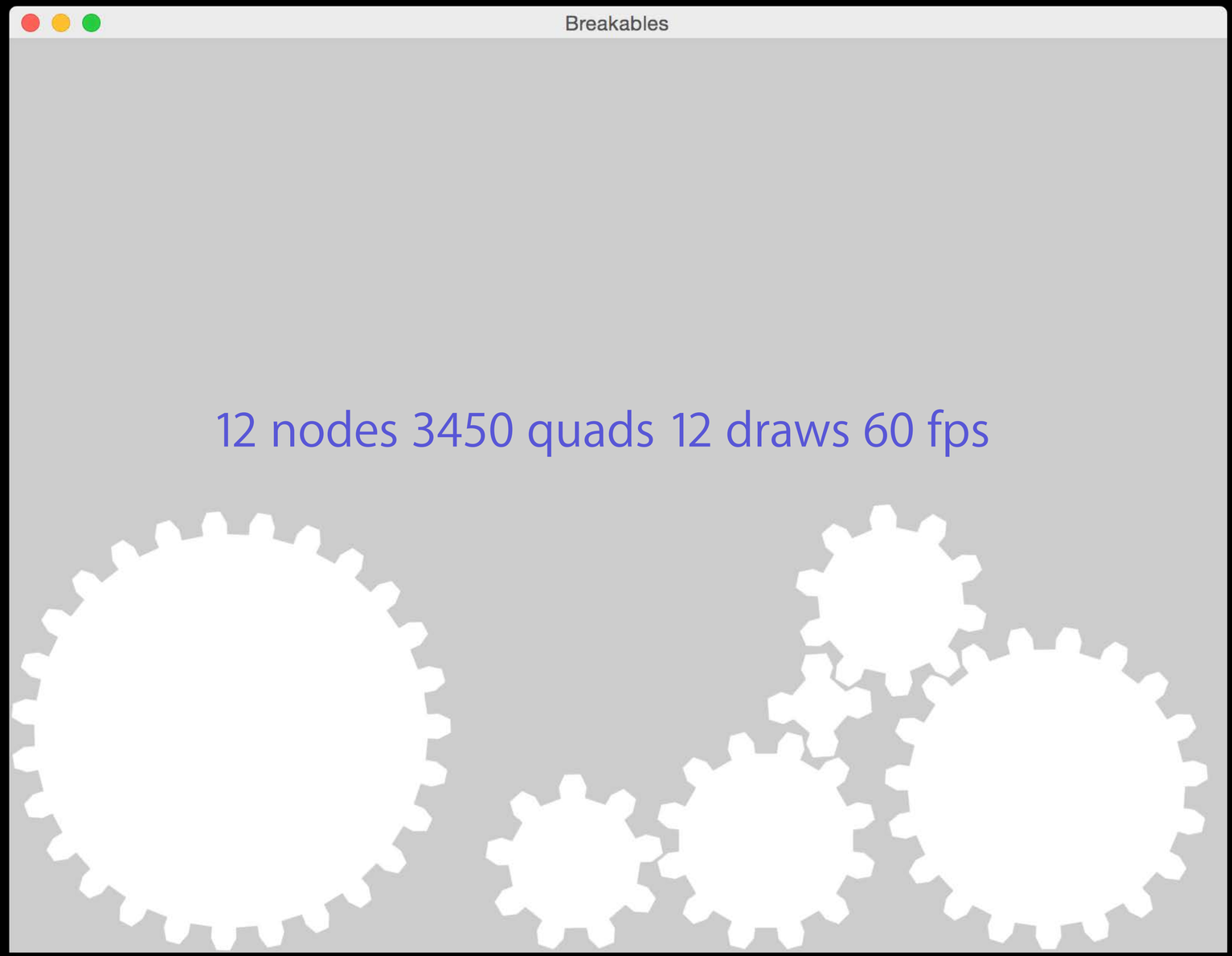
Tools

Tools to help evaluate graphics performance

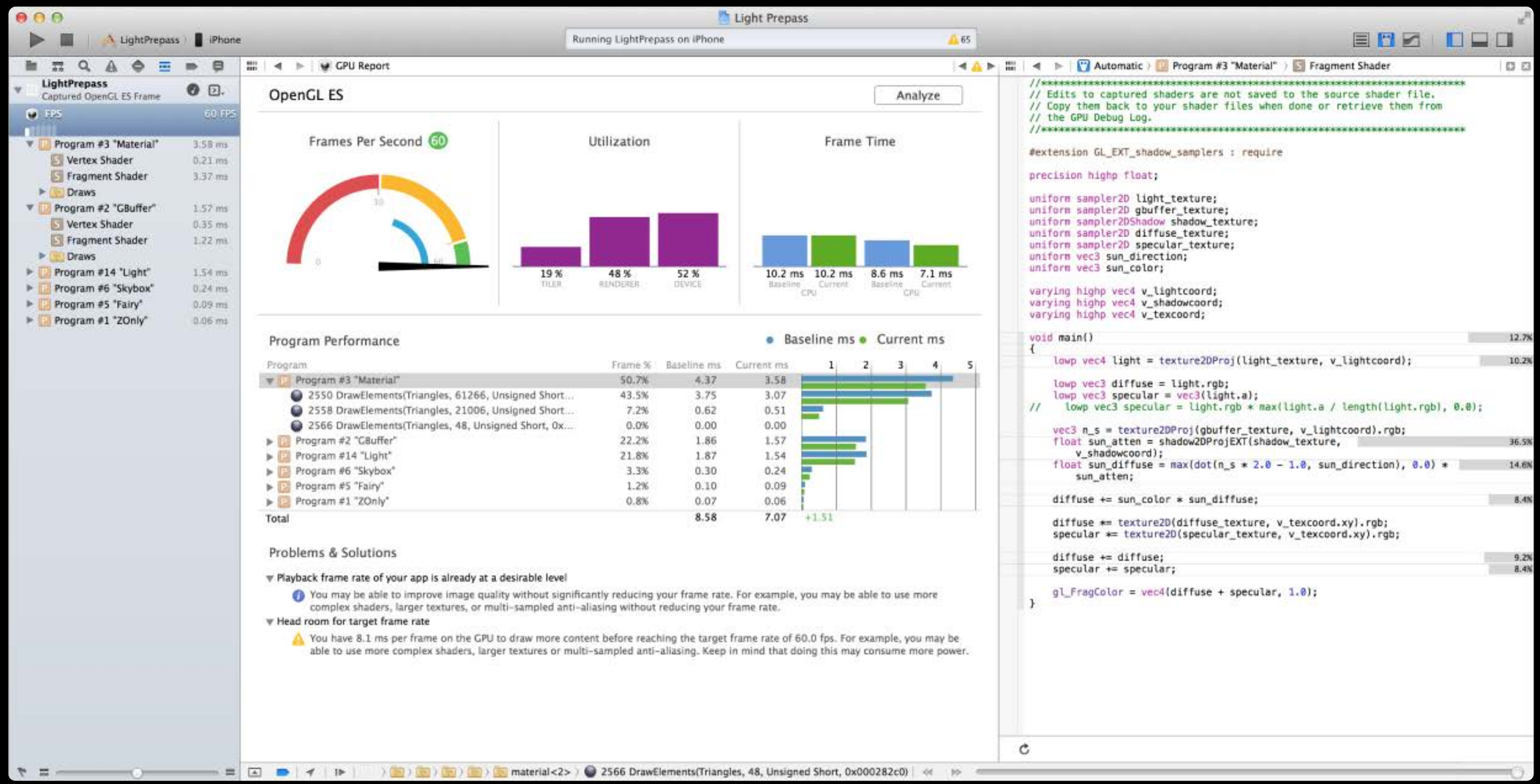
HUD flags on the view

```
@property (nonatomic) BOOL showsFPS;
@property (nonatomic) BOOL showsDrawCount;
@property (nonatomic) BOOL showsNodeCount;
@property (nonatomic) BOOL showsQuadCount;
```

Tools



Tools


Light Prepass
LightPrepass
iPhone
Running LightPrepass on iPhone
65
GPU Report
Automatic
Program #3 "Material"
Fragment Shader
LightPrepass
Captured OpenGL ES Frame
FPS
60 FPS
Program #3 "Material" 3.58 ms
Vertex Shader 0.21 ms
Fragment Shader 3.37 ms
Draws
Program #2 "GBuffer" 1.57 ms
Vertex Shader 0.35 ms
Fragment Shader 1.22 ms
Draws
Program #14 "Light" 1.54 ms
Program #6 "Skybox" 0.24 ms
Program #5 "Fairy" 0.09 ms
Program #1 "ZOnly" 0.06 ms
OpenGL ES
Analyze
Frames Per Second 60
Utilization
19 % TILER
48 % RENDERER
52 % DEVICE
Frame Time
10.2 ms Baseline
10.2 ms Current
8.6 ms Baseline
7.1 ms Current
CPU
CPU
Program Performance
Baseline ms
Current ms
Program
Frame %
Baseline ms
Current ms
Program #3 "Material" 50.7% 4.37 3.58
2550 DrawElements(Triangles, 61266, Unsigned Short... 43.5% 3.75 3.07
2558 DrawElements(Triangles, 21006, Unsigned Short... 7.2% 0.62 0.51
2566 DrawElements(Triangles, 48, Unsigned Short, 0x... 0.0% 0.00 0.00
Program #2 "GBuffer" 22.2% 1.86 1.57
Program #14 "Light" 21.8% 1.87 1.54
Program #6 "Skybox" 3.3% 0.30 0.24
Program #5 "Fairy" 1.2% 0.10 0.09
Program #1 "ZOnly" 0.8% 0.07 0.06
Total 8.58 7.07 +1.51
Problems & Solutions
Playback frame rate of your app is already at a desirable level
You may be able to improve image quality without significantly reducing your frame rate. For example, you may be able to use more complex shaders, larger textures, or multi-sampled anti-aliasing without reducing your frame rate.
Head room for target frame rate
You have 8.1 ms per frame on the GPU to draw more content before reaching the target frame rate of 60.0 fps. For example, you may be able to use more complex shaders, larger textures or multi-sampled anti-aliasing. Keep in mind that doing this may consume more power.
//**********
// Edits to captured shaders are not saved to the source shader file.
// Copy them back to your shader files when done or retrieve them from
// the GPU Debug Log.
//**********
#extension GL_EXT_shadow_samplers : require
precision highp float;
uniform sampler2D light_texture;
uniform sampler2D gbuffer_texture;
uniform sampler2DShadow shadow_texture;
uniform sampler2D diffuse_texture;
uniform sampler2D specular_texture;
uniform vec3 sun_direction;
uniform vec3 sun_color;
varying highp vec4 v_lightcoord;
varying highp vec4 v_shadowcoord;
varying highp vec4 v_texcoord;
void main()
12.7%
{
lowp vec4 light = texture2DProj(light_texture, v_lightcoord);
10.2%
lowp vec3 diffuse = light.rgb;
lowp vec3 specular = vec3(light.a);
// lowp vec3 specular = light.rgb * max(light.a / length(light.rgb), 0.0);
vec3 n_s = texture2DProj(gbuffer_texture, v_lightcoord).rgb;
float sun_atten = shadow2DProjEXT(shadow_texture,
v_shadowcoord);
36.5%
float sun_diffuse = max(dot(n_s * 2.0 - 1.0, sun_direction), 0.0) *
sun_atten;
14.6%
diffuse += sun_color * sun_diffuse;
8.4%
diffuse *= texture2D(diffuse_texture, v_texcoord.xy).rgb;
specular *= texture2D(specular_texture, v_texcoord.xy).rgb;
diffuse += diffuse;
9.2%
specular += specular;
8.4%
gl_FragColor = vec4(diffuse + specular, 1.0);
}
material<2>
2566 DrawElements(Triangles, 48, Unsigned Short, 0x000282c0)

Drawing Performance

Key insights

Drawing Performance

Key insights

Compose scenes as layers

- Give objects a common Z value per layer
- Place overlapping objects in different layers

```
view.ignoreSiblingOrder = YES;
```

Share shaders, textures, and procedural textures

Keep blend modes on the same Z layer

Use HUD features, profilers

Actions and Constraints

Actions and Constraints

Overview

Actions and Constraints

Overview

Actions are commands to the high efficiency SpriteKit engine

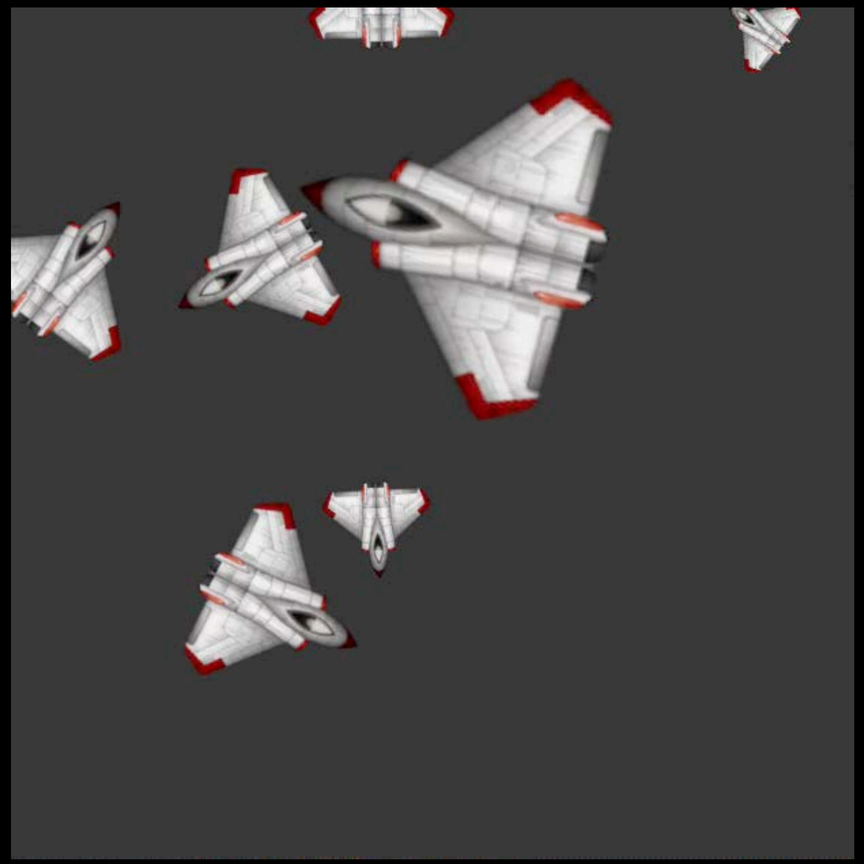
```
[node runAction: [SKAction rotateByAngle:M_PI duration:1.0] ];
```

One line creation

Chain them, group them sequence them, reuse them

Actions and Constraints

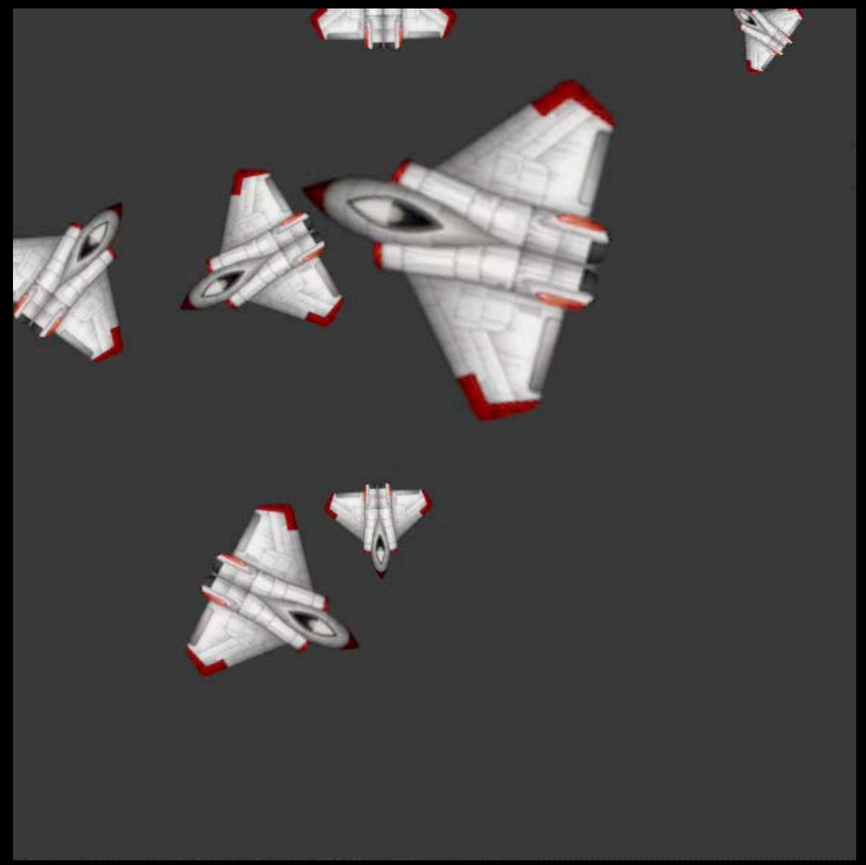
Overview



Actions and Constraints

Overview

```
[SKAction rotateByAngle:M_PI duration:1.0];
[SKAction moveTo:aCGPoint duration:1.0];
[SKAction scaleBy:2.0 duration:1.0];
```



Actions and Constraints

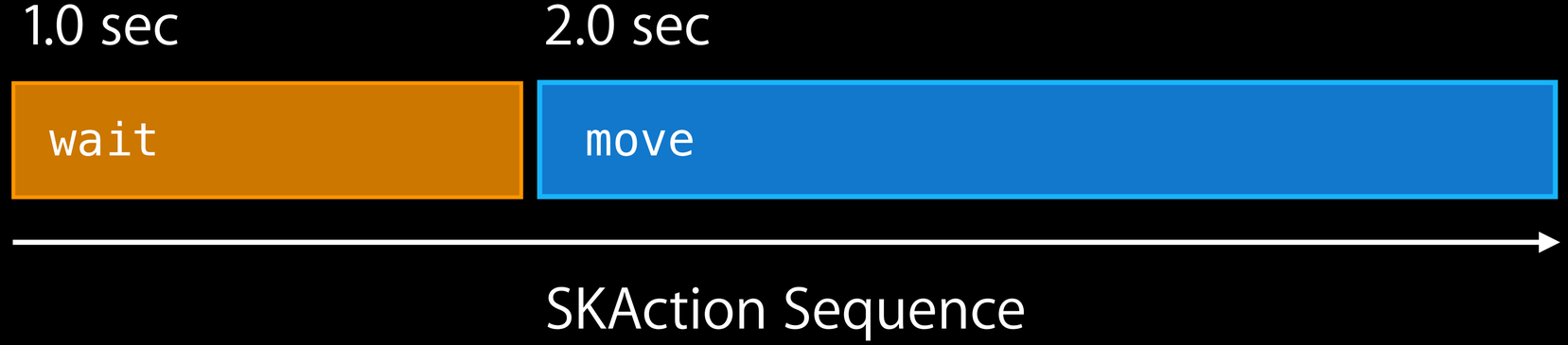
Sequence

```
[myNode runAction: [SKAction sequence:@[wait, move]] ];
```

Actions and Constraints

Sequence

```
[myNode runAction: [SKAction sequence:@[wait, move]] ];
```



Actions and Constraints

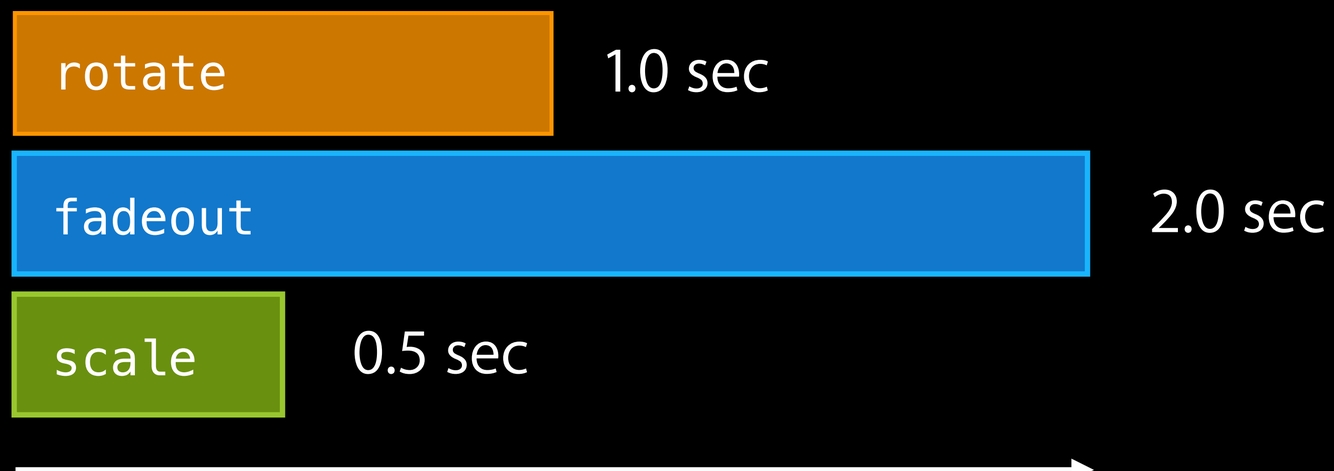
Sequence

```
[myNode runAction: [SKAction group:@[rotate, fadeout, scale]] ];
```

Actions and Constraints

Sequence

```
[myNode runAction: [SKAction group:@[rotate, fadeout, scale]] ];
```



Actions and Constraints

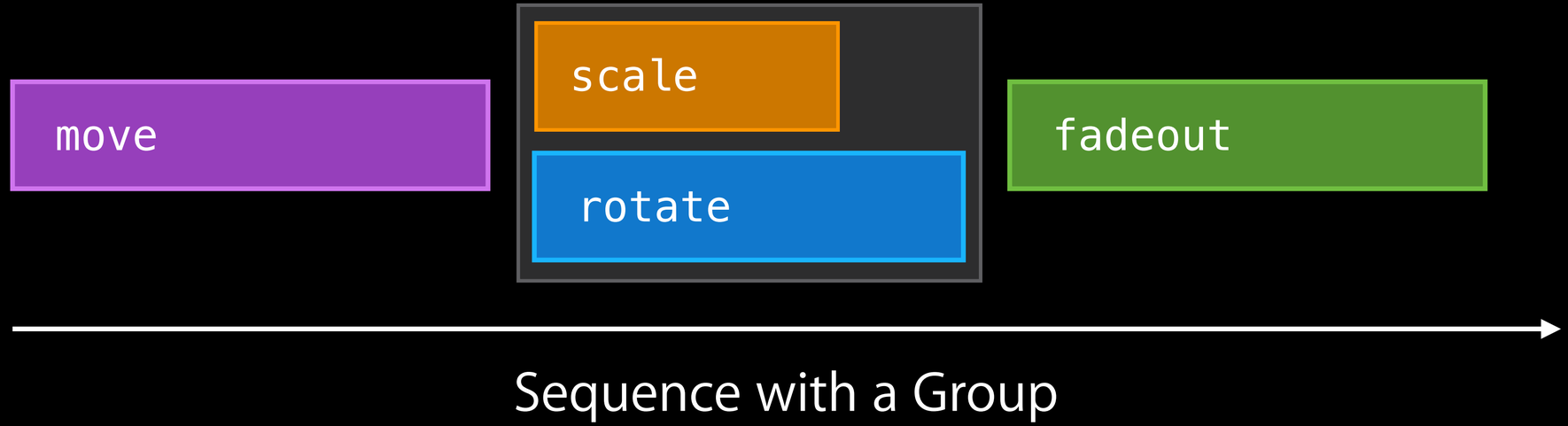
Sequence

```
SKAction *group = [SKAction group:@[scale, rotate]];

[myNode runAction: [SKAction sequence:@[move, group, fadeout]] ];
```

Actions and Constraints

Sequence

```
SKAction *group = [SKAction group:@[scale, rotate]];

[myNode runAction: [SKAction sequence:@[move, group, fadeout]] ];
```



Sequence with a Group

Actions and Constraints

Overview

There’s a huge catalog available

Actions and Constraints

Overview

There's a huge catalog available

moveByX: y: duration:

moveTo: duration:

moveToX: duration:

moveToY: duration:

rotateByAngle: duration:

rotateToAngle: duration:

scaleXTo: duration:

scaleYTo: duration:

speedBy: duration:

speedTo: duration:

scaleBy: duration:

scaleXBy: y: duration:

followPath: duration:

waitForDuration:

scaleTo: duration:

scaleXTo: y: duration:

sequence:

group:

setTexture:

runBlock:

runBlock: queue:

removeFromParent

performSelector: onTarget:

resizeByWidth: height: duration:

resizeToWidth: height: duration:

resizeToWidth: duration:

playSoundFileNamed: waitForCompletion:

colorizeWithColor: colorBlendFactor:

colorizeWithColorBlendFactor: duration:

resizeToHeight: duration:

repeatAction: count:

repeatActionForever:

fadeInWithDuration:

fadeOutWithDuration:

fadeAlphaBy: duration:

fadeAlphaTo: duration:

animateWithTextures: timePerFrame:

animateWithTextures: timePerFrame: resize:

playSoundFileNamed: waitForCompletion:

colorizeWithColor: colorBlendFactor:

colorizeWithColorBlendFactor: duration:

followPath: duration:

waitForDuration: withRange:

runAction: onChildWithName:

followPath: asOffset: orientToPath:

waitForDuration:

waitForDuration: withRange:

runAction: onChildWithName:

customActionWithDuration: actionBlock:
resizeToHeight: duration:

repeatAction: count:

repeatActionForever:

fadeInWithDuration:

fadeOutWithDuration:

fadeAlphaBy: duration:

fadeAlphaTo: duration:

animateWithTextures: timePerFrame:

animateWithTextures: timePerFrame: resize:

customActionWithDuration: actionBlock:

Actions and Constraints

Use actions instead of logic in update

Actions and Constraints

Use actions instead of logic in update

Eliminate animation code from your update

Actions and constraints can do the job

Actions and Constraints

Use actions instead of logic in update

Eliminate animation code from your update

Actions and constraints can do the job

Actions and Constraints

Example

Actions and Constraints

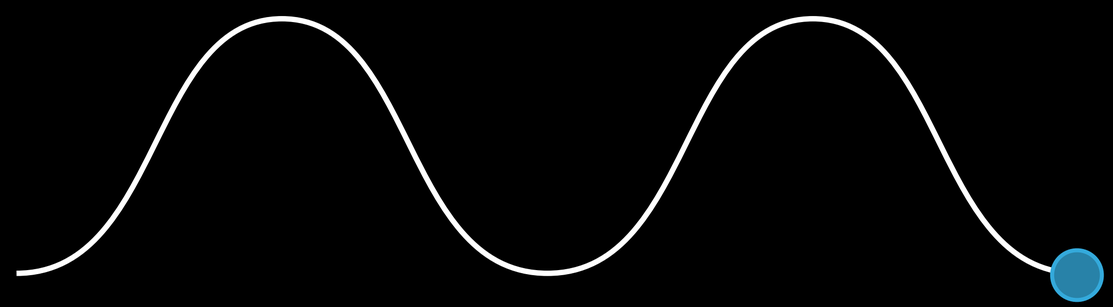
Example

followPath has new functionality

Leverage SKShapeNode's new Spline Point shape

```
CGPathRef p = [SKShapeNode shapeWithSplinePoints:points]
[node runAction:[SKAction followpath:p];
```

Path will be followed at constant velocity



Actions and Constraints

Use actions instead of logic in update

SKConstraints

- OrientToNode

Actions and Constraints

Use actions instead of logic in update

SKConstraints

- OrientToNode

Actions and Constraints

Re-use actions

Actions and Constraints

Re-use actions

Build once

Actions are copy on add, perfect for re-use

Actions run when a node is added to scene

- Make a spaceship with an entry action
- Copy the spaceship and add it to the scene
- The entry action will then run automatically

Actions

Named actions

Actions

Named actions

Create an action with a named key

```
[sprite runAction:[SKAction moveTo:CGPointMake(x, y) duration:1]
          withKey:@"move"];
```

Override an action in progress easily using the same key

Cancel an action

```
[sprite removeActionForKey:@"move"];
```

Physics Best Practices

Physics Overview

Rigid bodies

Physics Overview

Rigid bodies

Rigid body dynamics

- Bouncing, falling, rolling, sliding

Collision handling, contact delegates

Fields

Physics Performance

Minimize computation

Physics Performance

Minimize computation

Static objects are inexpensive, even if the shape is complex

```
physicsBody.dynamic = NO
```

Physics Performance

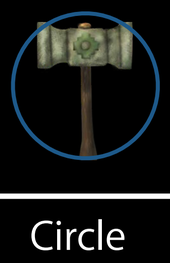
Pick the right shape

Different shapes have different costs

Physics Performance

Pick the right shape

Different shapes have different costs



Efficient — Convenient

Circle

Physics Performance

Pick the right shape

Different shapes have different costs

Efficient — Convenient

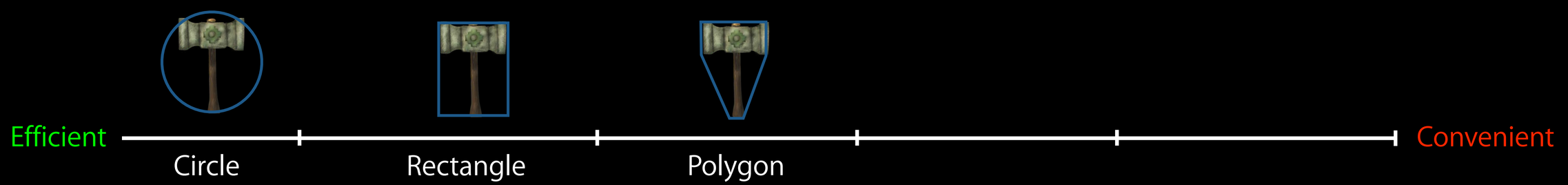
Circle

Rectangle

Physics Performance

Pick the right shape

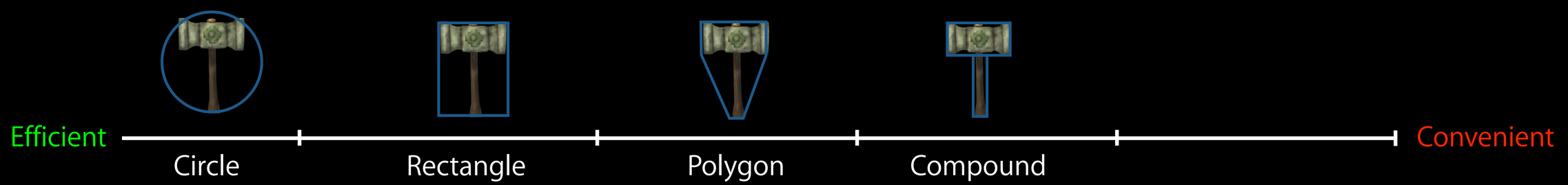
Different shapes have different costs



Physics Performance

Pick the right shape

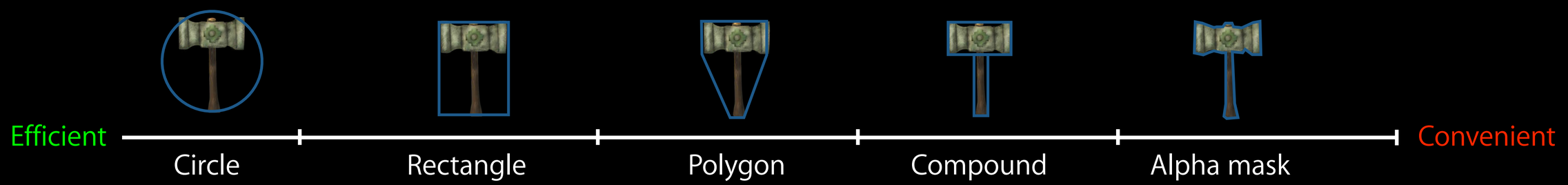
Different shapes have different costs



Physics Performance

Pick the right shape

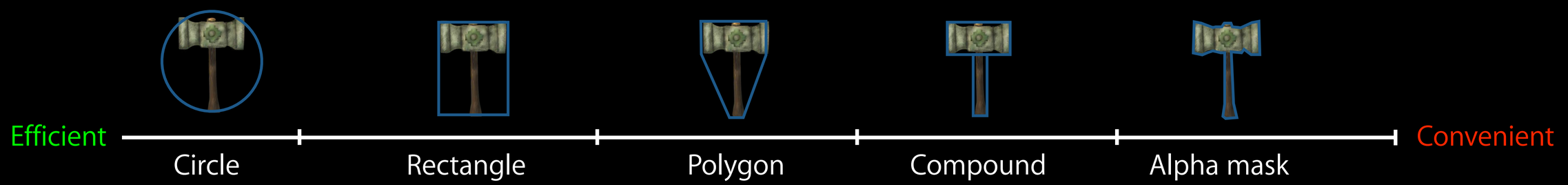
Different shapes have different costs



Physics Performance

Pick the right shape

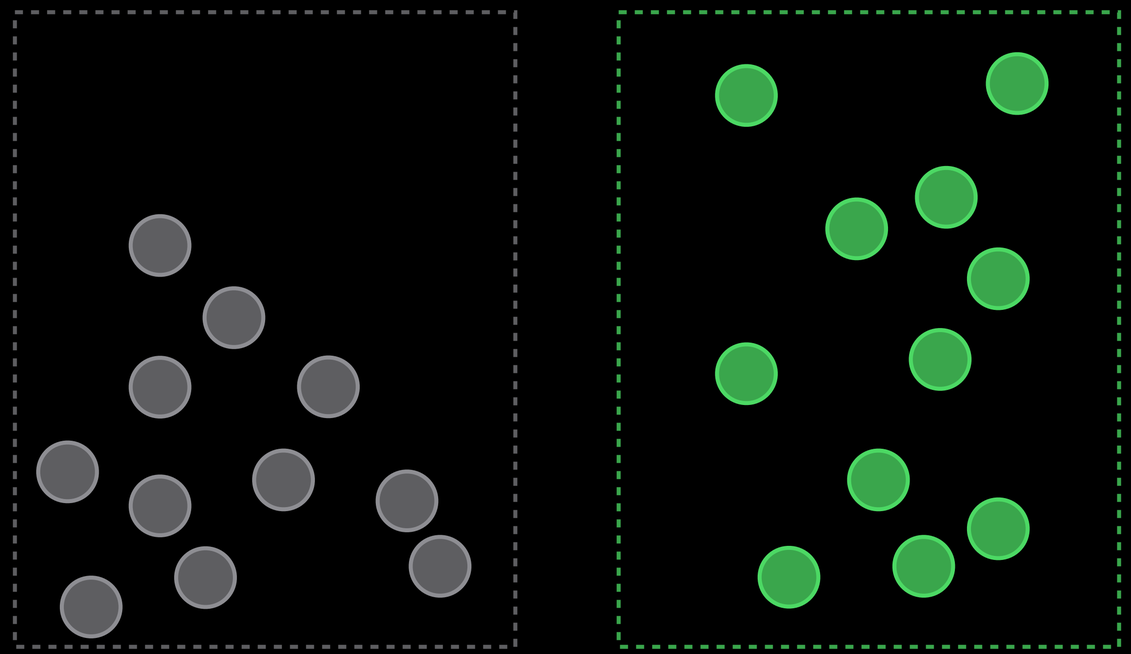
Different shapes have different costs



Pick the cheapest representation that serves your game

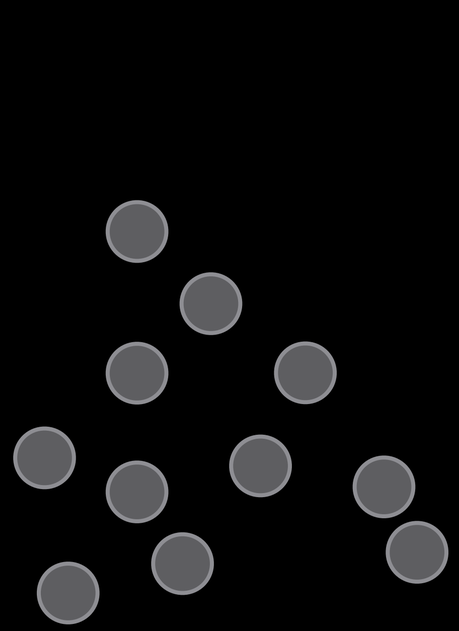
Physics Performance

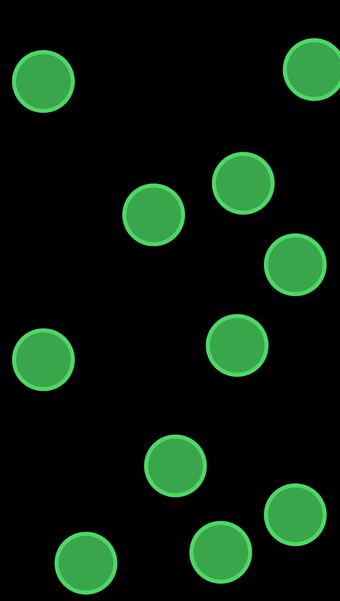
Use Collision Masks to group objects for performance



Physics Performance

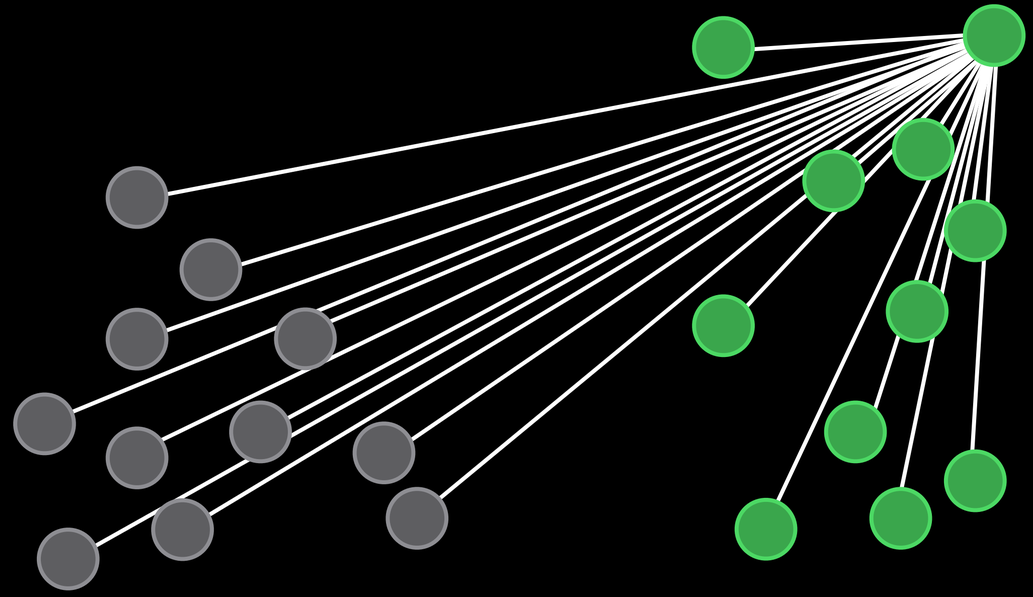
Use Collision Masks to group objects for performance





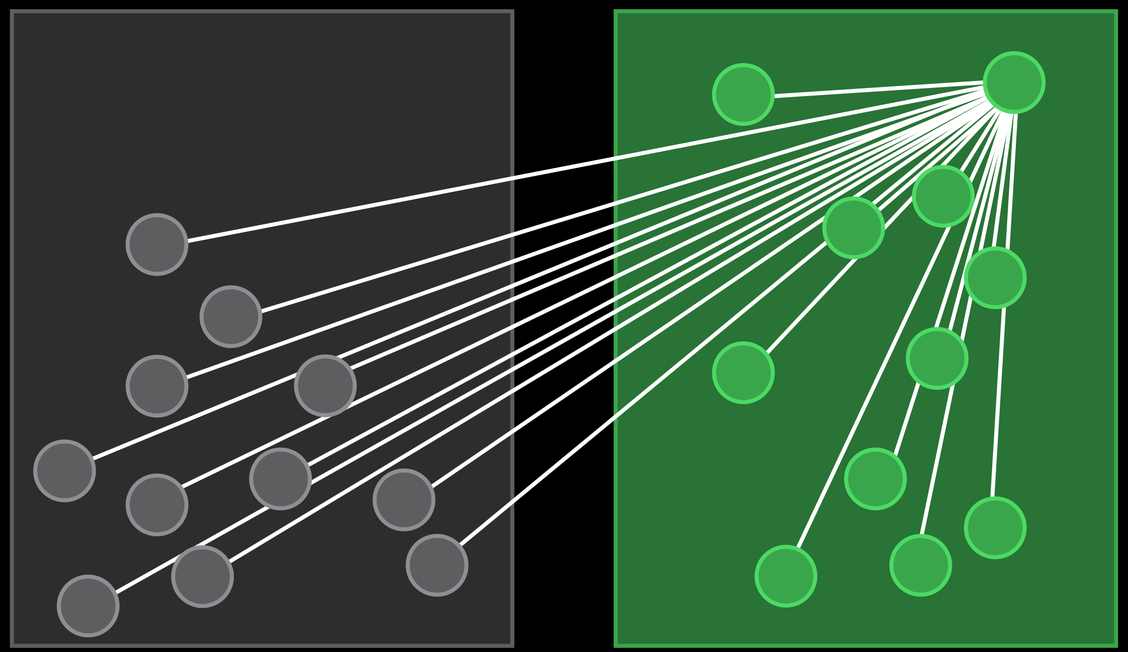
Physics Performance

Use Collision Masks to group objects for performance



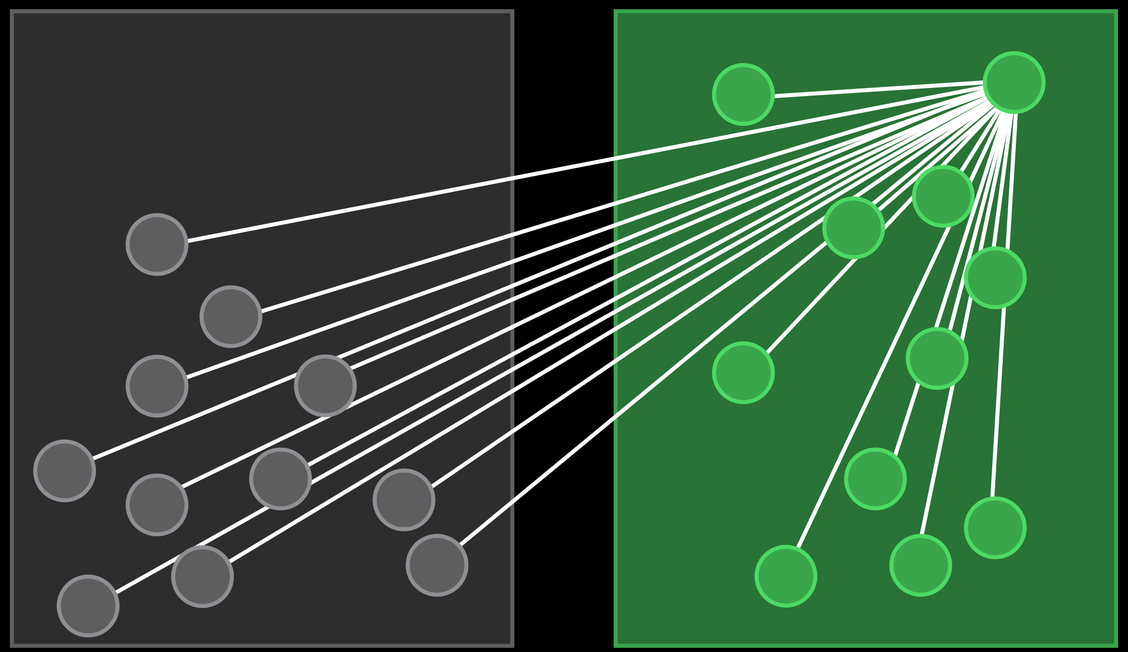
Physics Performance

Use Collision Masks to group objects for performance



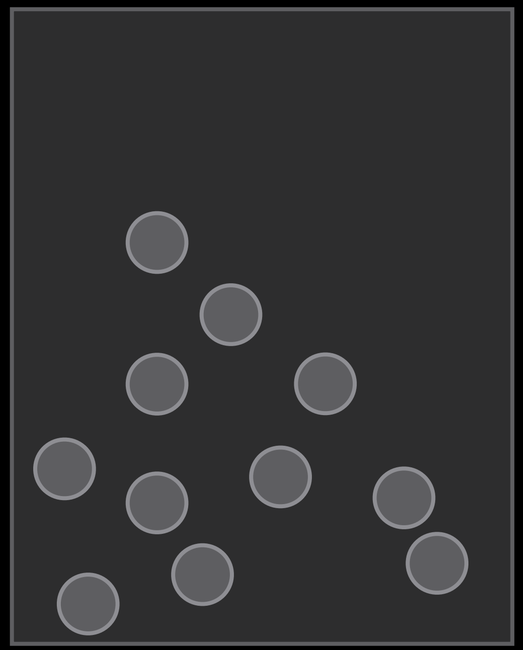
Physics Performance

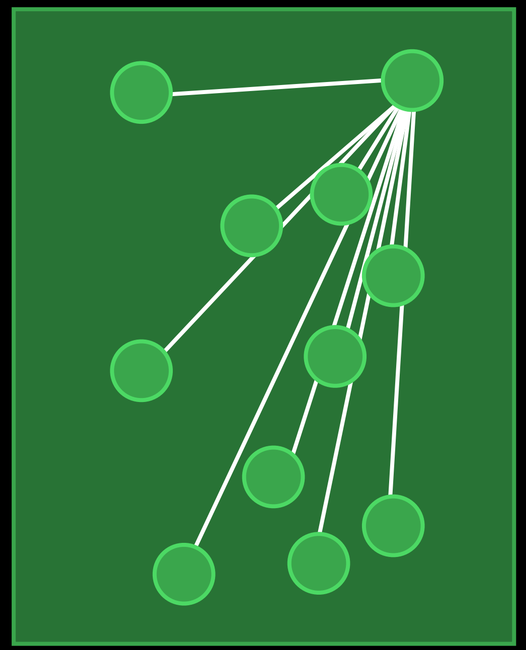
Use Collision Masks to group objects for performance



Physics Performance

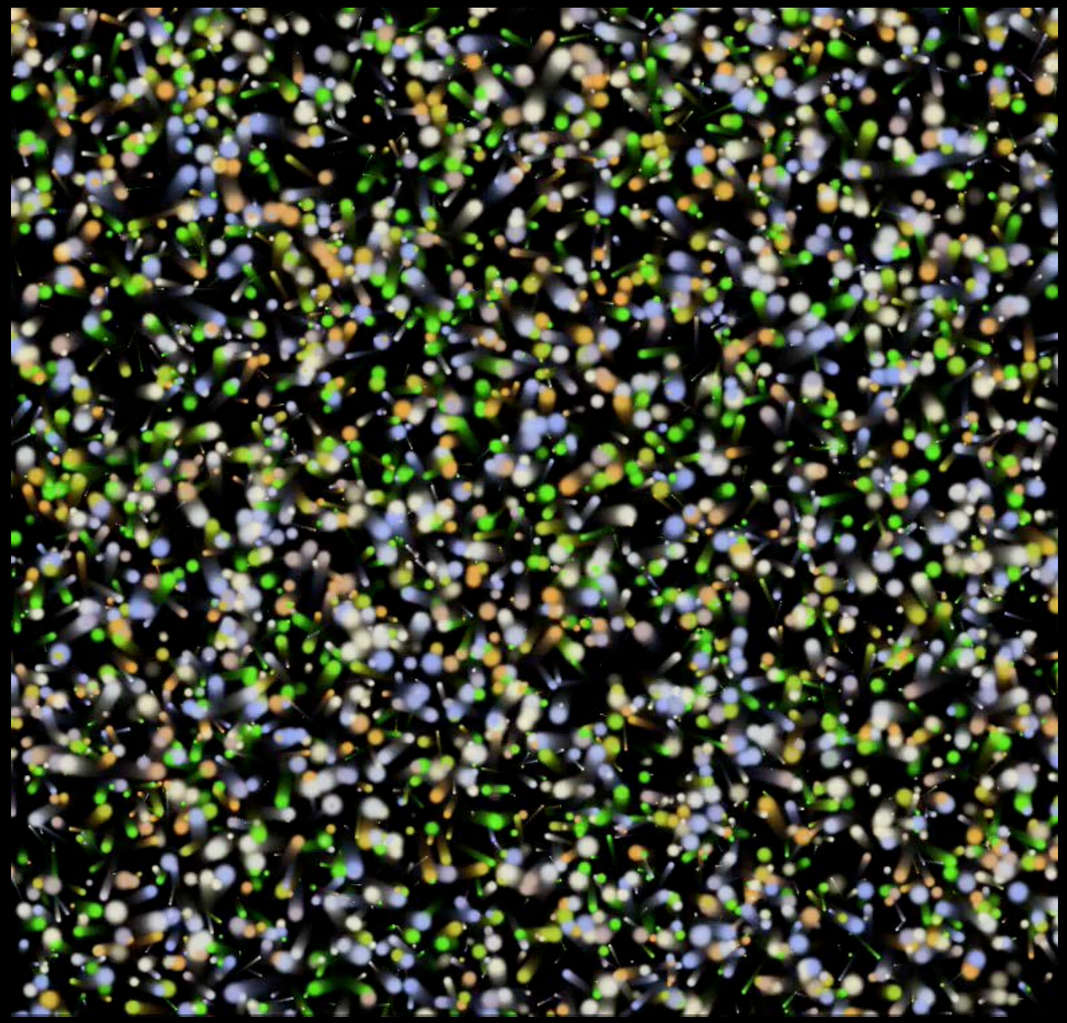
Use Collision Masks to group objects for performance





Physics Overview

Force fields



Physics Overview

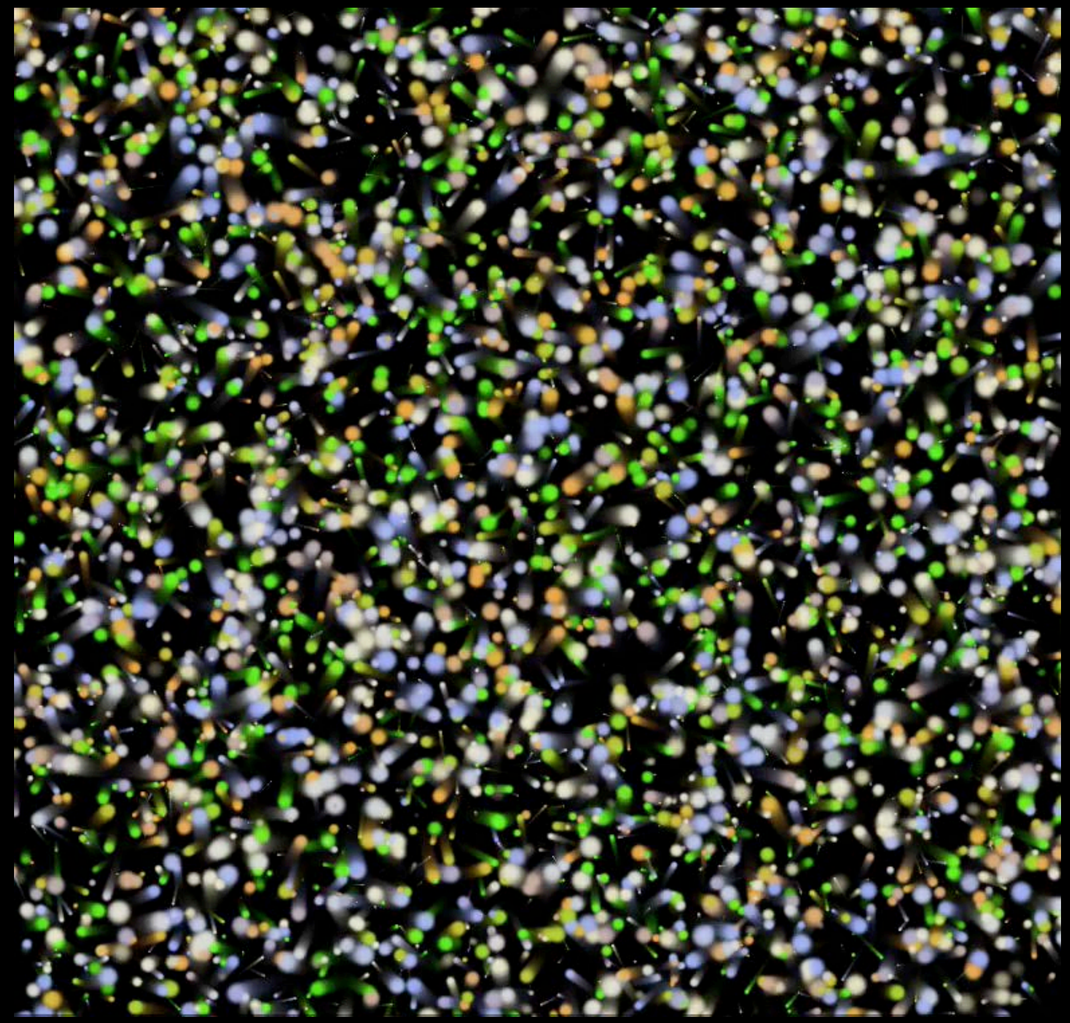
Force fields

Fields

Variety of types

Inexpensive to compute

Use actions to fade in and out



Physics Overview

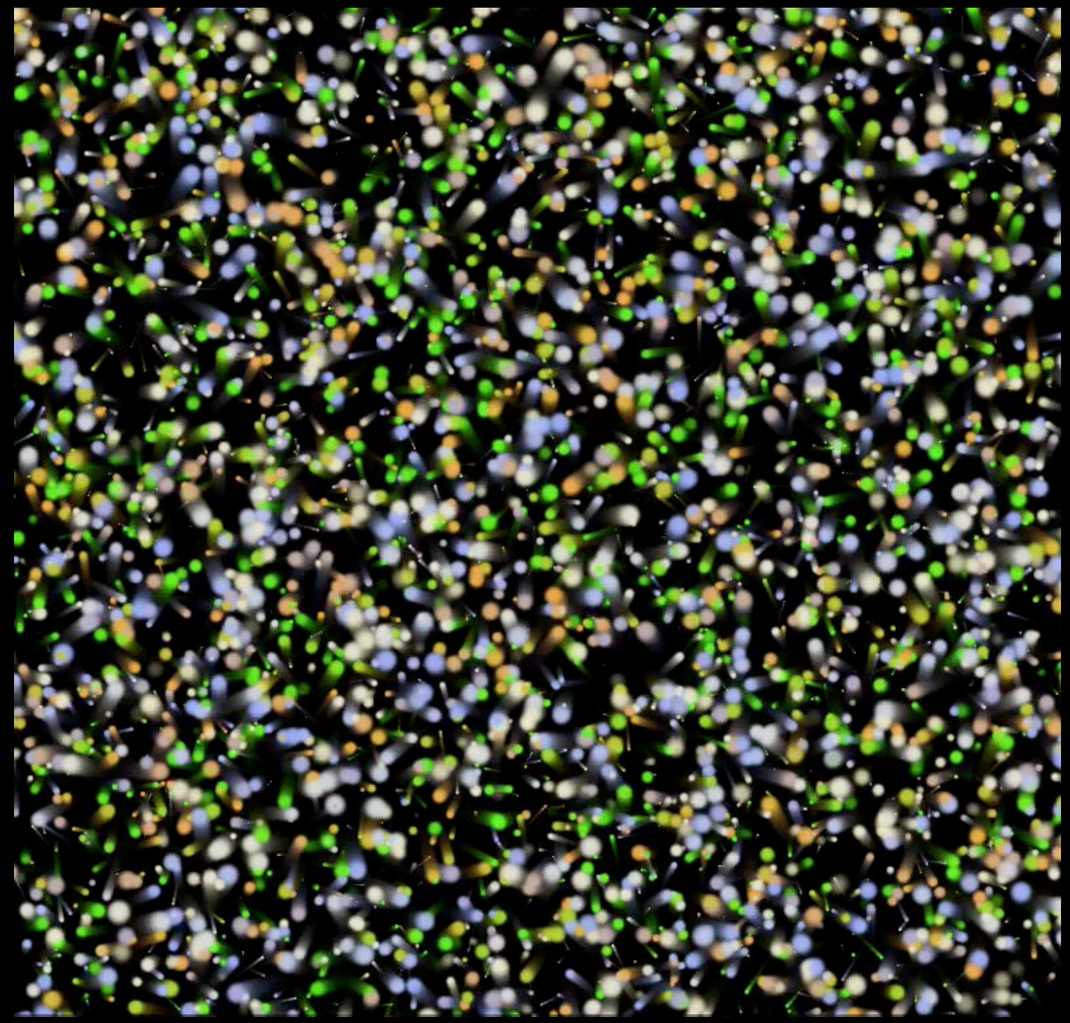
Force fields

Fields

Variety of types

Inexpensive to compute

Use actions to fade in and out



Physics Overview

Force fields

Physics Overview

Force fields

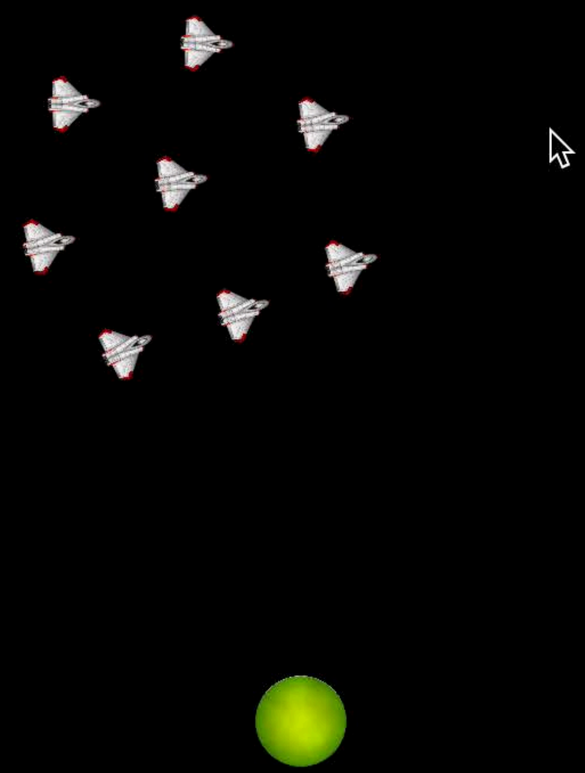
Physics

Physics

```
float dx = targetX - ship->sprite.position.x;
float dy = targetY - ship->sprite.position.y;
float dist = dx*dx+dy*dy;
if (fabsf(dist) > 0.01f) {
    dist = sqrtf(dist);
    float vx = ship->sprite.physicsBody.velocity.dx;
    float vy = ship->sprite.physicsBody.velocity.dy;
    ship->sprite.zRotation = atan2f(vy, vx) - 0.5f * M_PI;
    dx = (dx - vx) * dist * 0.0001f;
    dy = (dy - vy) * dist * 0.0001f;
    [ship->sprite.physicsBody applyImpulse:CGVectorMake(dx, dy)];
}
```

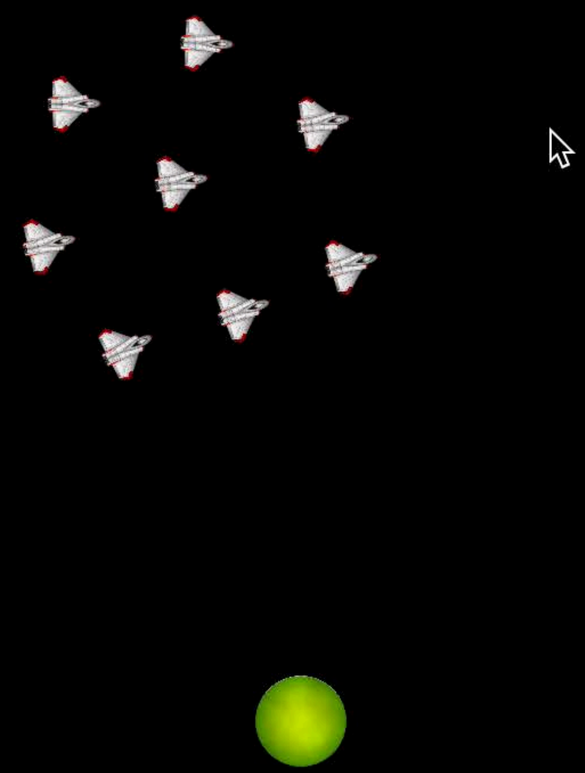
Physics

Field debug drawing



Physics

Field debug drawing



Physics Best Practices

Take aways

Physics Best Practices

Take aways

Choose the cheapest appropriate rigid body

Separate groups with masks

Fields can replace traditional update logic

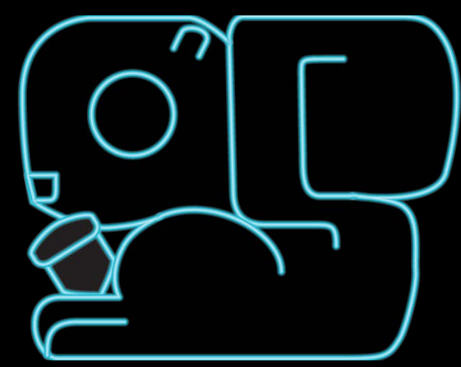
Use the debug drawing

Shapes Best Practices

Sprites vs. Shapes

Sprites for Bitmap art

Shapes for Vector art

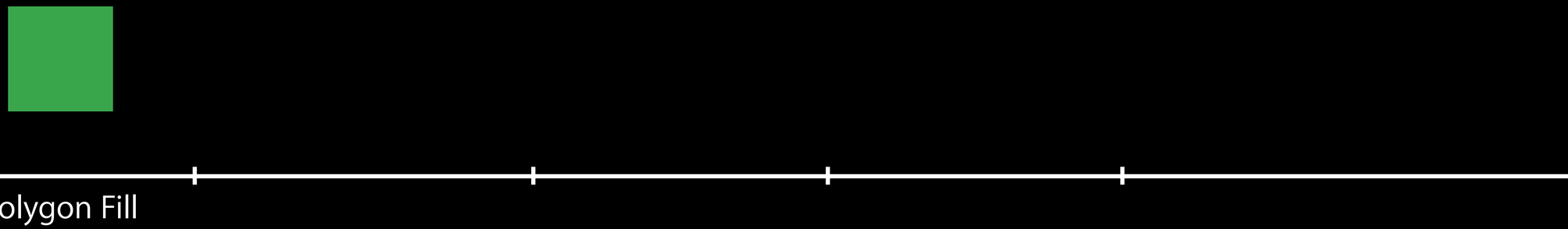


Performance

SKShapeNode

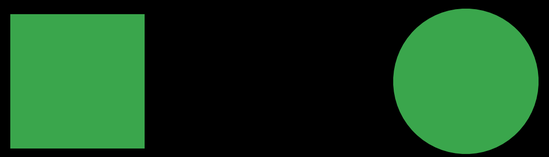
Performance

SKShapeNode



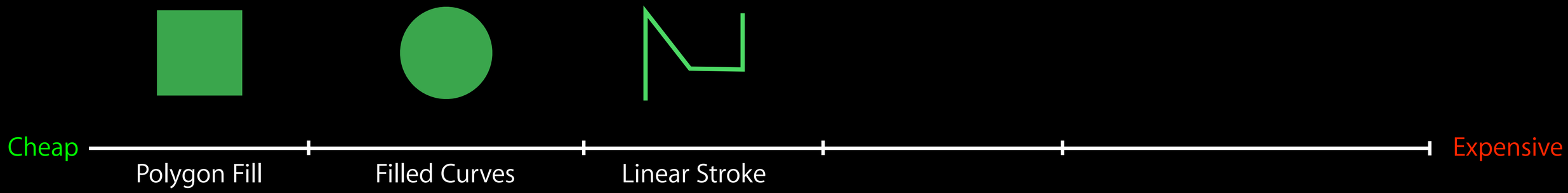
Performance

SKShapeNode



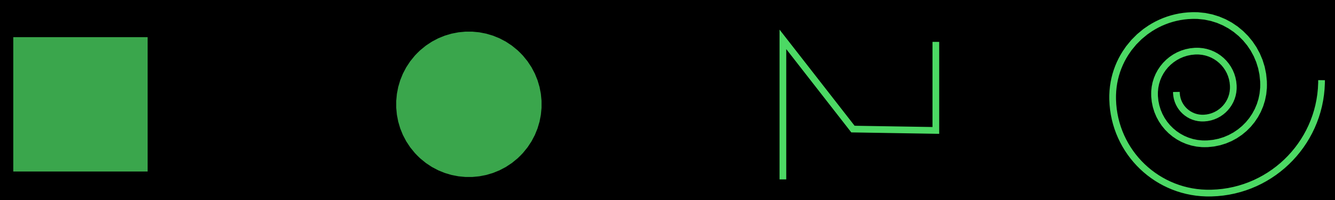
Performance

SKShapeNode



Performance

SKShapeNode

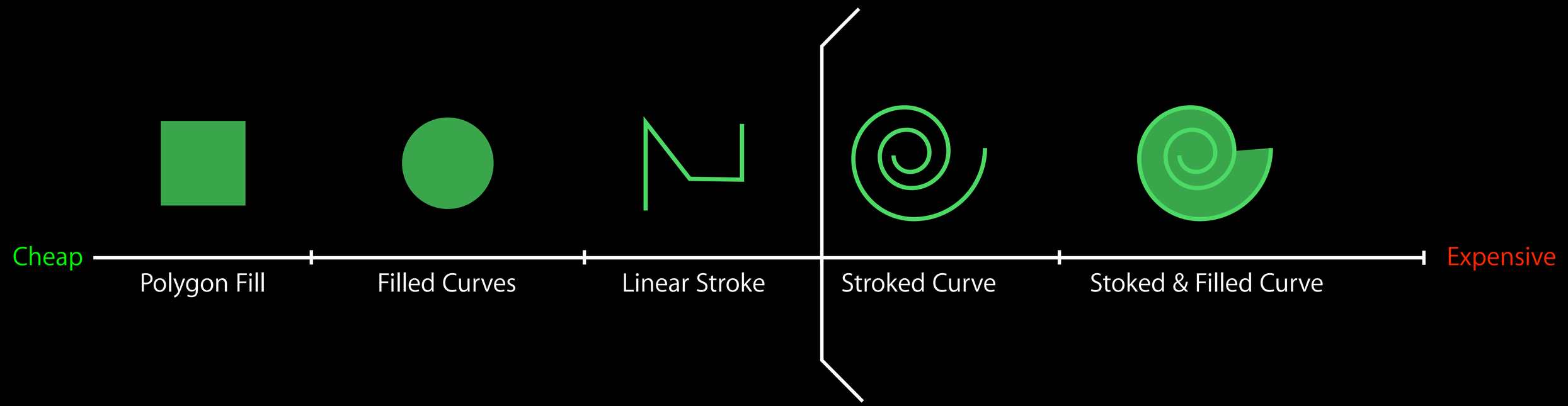


Performance

SKShapeNode

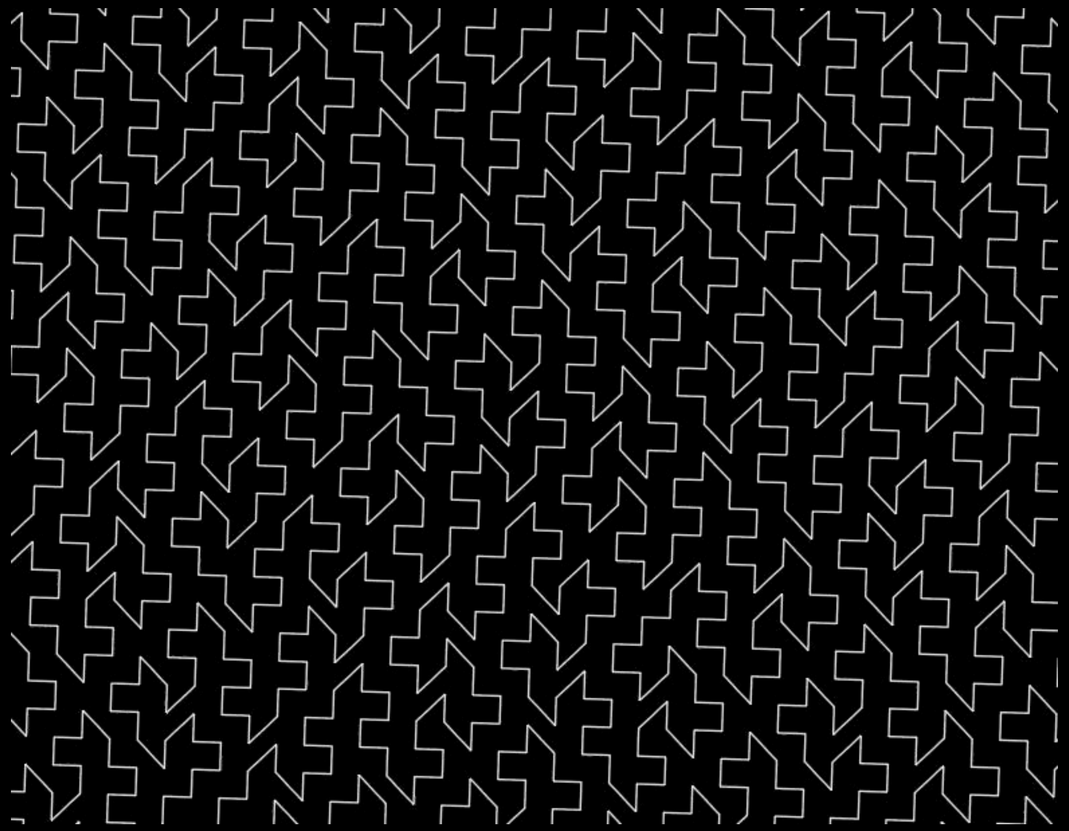
Performance

SKShapeNode



Performance

Piecewise linear stroke

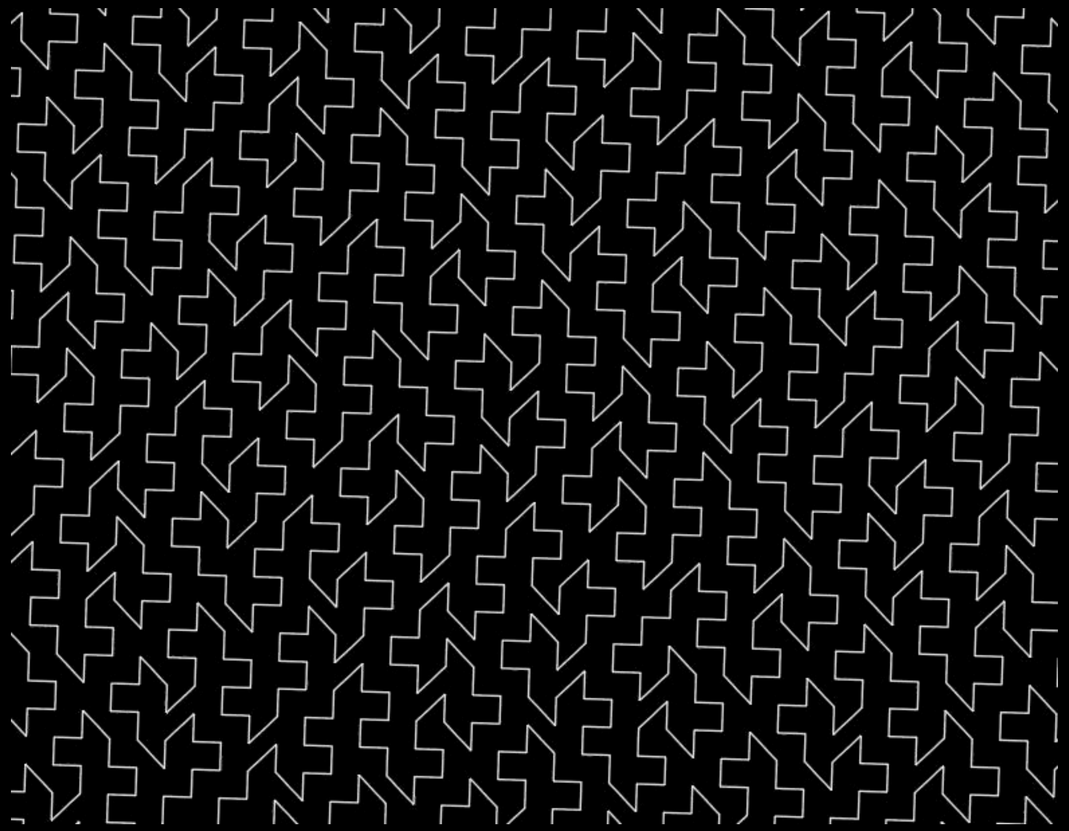


Performance

Piecewise linear stroke

Piecewise linear shapes are very cheap

Even for complex paths

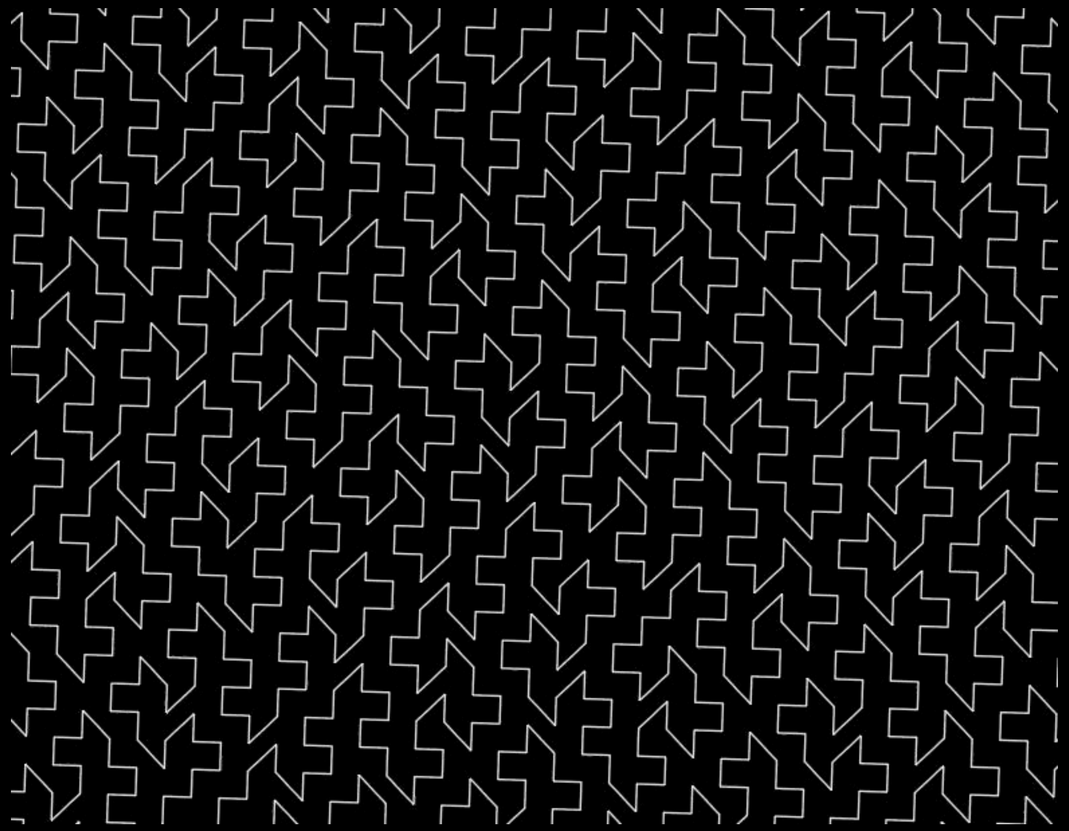


Performance

Piecewise linear stroke

Piecewise linear shapes are very cheap

Even for complex paths



Effects Best Practices

SKEffectNodes

Effects Best Practices

SKEffectNodes

SKEffectNodes are for offscreen rendering

- Use sparingly

Powerful CoreImage support

Effects Best Practices

SKEffectNodes

SKEffectNodes are for offscreen rendering

- Use sparingly

Powerful CoreImage support

Utilize SKShaders when no offscreen pass is needed

Effects Best Practices

SKEffectNodes

SKEffectNodes are for offscreen rendering

- Use sparingly

Powerful CoreImage support

Utilize SKShaders when no offscreen pass is needed

Rasterize complex effects if they don’t change too much

```
effect.shouldRasterize = YES;
```

Effects Best Practices

Special effects

Bake static special effects to a texture

- From any SKNode subtree

```
SKTexture *texture = [myView textureFromNode:node size:size];
```

- By applying a CIFilter

```
SKTexture *texture = [myTexture textureByApplyingCIFilter:filter];
```

Lighting Best Practices

Lighting

SKLightNode

Lighting cost proportional to lit pixels

Ambient light is free

Optimize with bit masks

Lighting

SKLightNode

Maximum eight lights per individual sprite

Normal maps are cheap

Reuse your maps!

Lighting

SKLightNode

Shadows cost proportional to the number of lights

Lighting

SKLightNode

Shadows cost proportional to the number of lights

The ideal is a few lights, and fewer shadows

Performance Best Practices

Summary

Performance Best Practices

Summary

Drawing performance

Actions and constraints

Physics

Shapes

Effects

Lighting

SpriteKit Best Practices

Agenda summary

SpriteKit Best Practices

Agenda summary

Scalability best practices

Game structure best practices

Performance best practices

More Information

Allan Schafer
Game Technologies Evangelist
aschaffer@apple.com

Filip Iliescu
Graphics and Media Evangelist
filiescu@apple.com

Documentation
SpriteKit Programming Guide
https://developer.apple.com/spritekitprogrammingguide

http://devforums.apple.com

WWDC14