Graphics and Games

#WWDC14

What's New in SpriteKit

Session 606

Norman Wang

Game Technologies

SpriteKit

Shaders

Lighting and Shadows

New Physics

Integration with SceneKit

Tools

Improvements

Shaders

Lighting and Shadows

New Physics

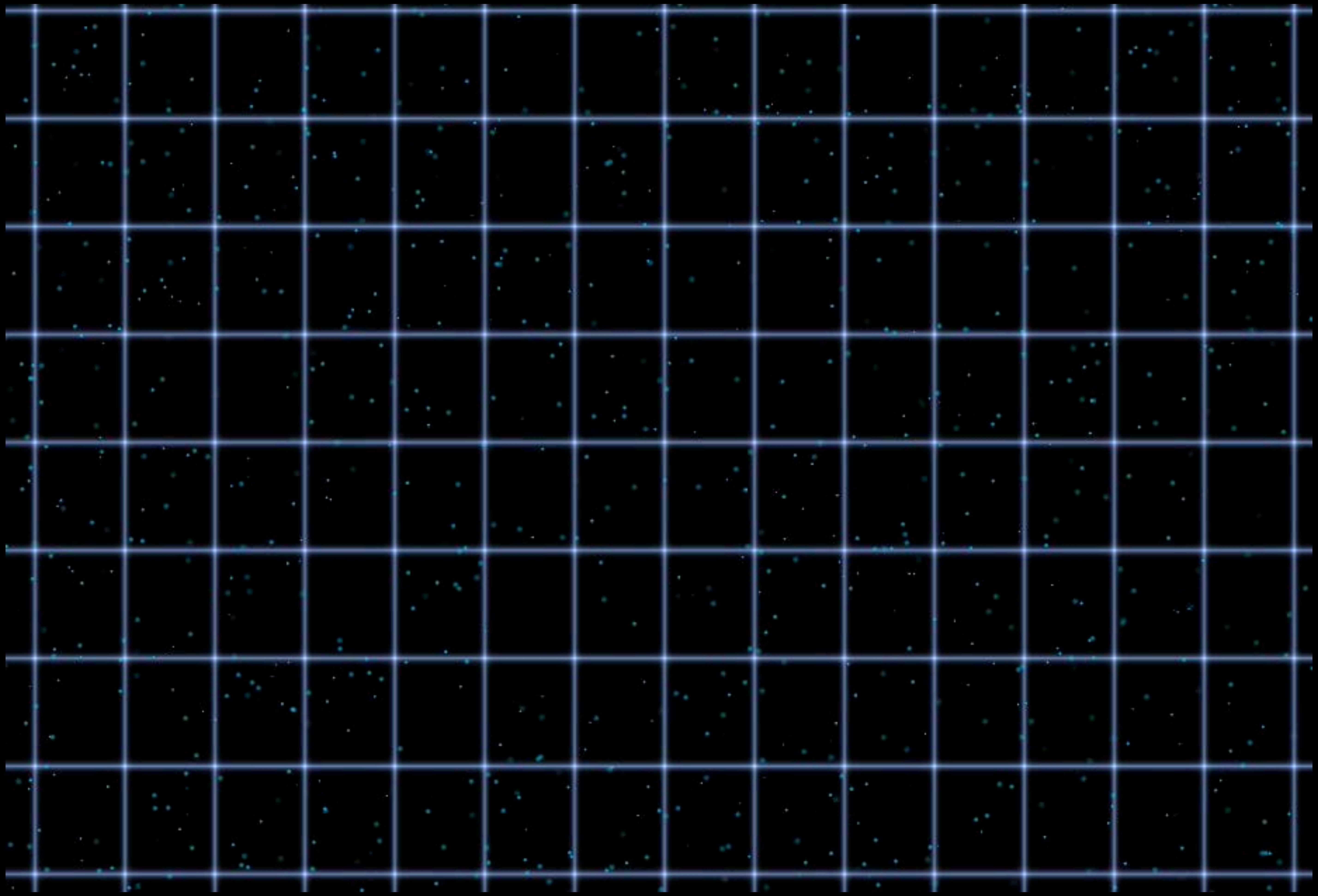
Integration with SceneKit

Tools

Improvements

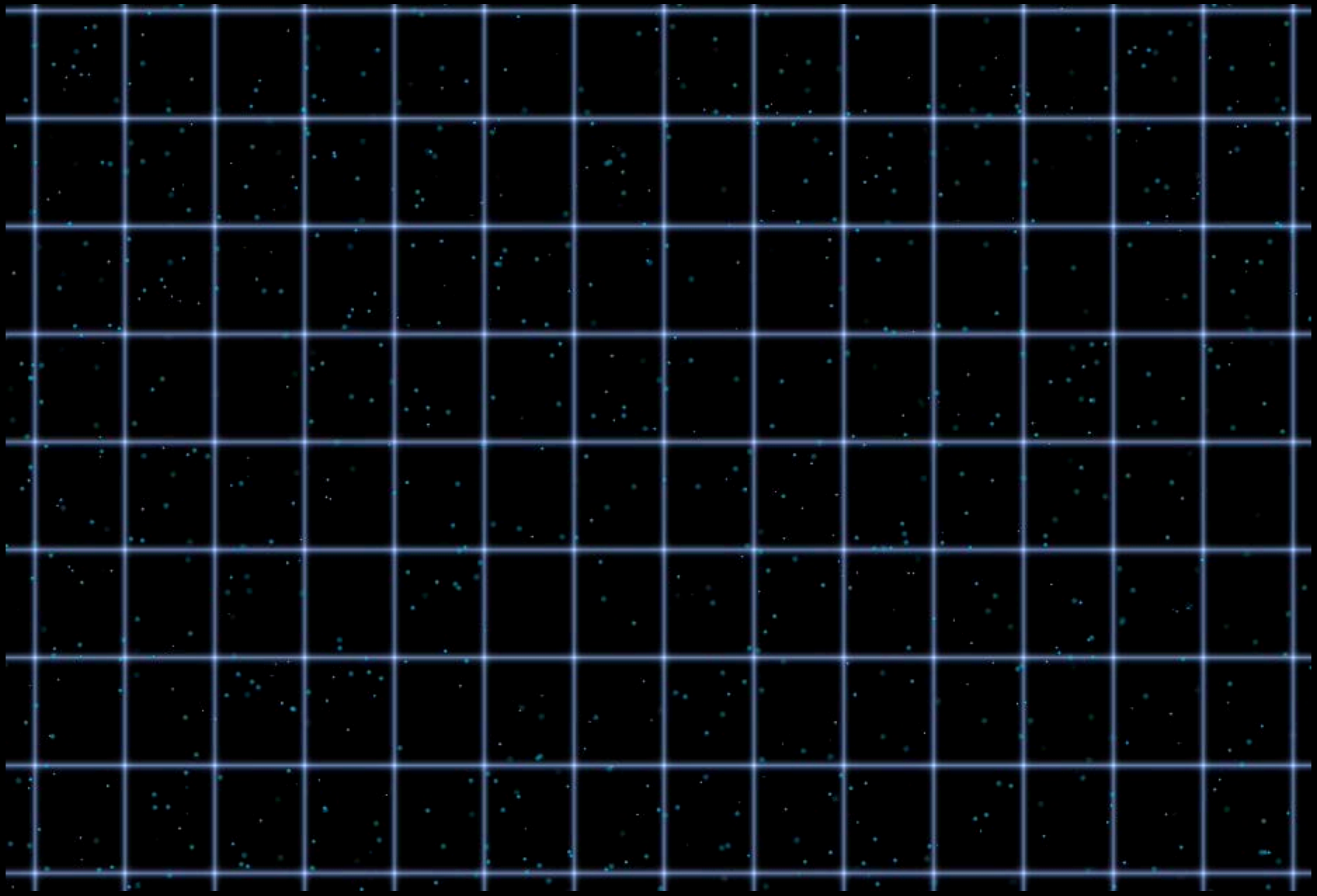
Shaders

Introduction



Shaders

Introduction



Shaders

Overview

Shaders customize the way things are drawn in a game

Programmed using a C-like language

Powerful tools that can produce a wide variety of effects

Useful to add new effects to existing games

Shaders

SKShaders object

Holds a custom OpenGL ES fragment shader

Deploy to both OS X and iOS

Supported node types

- SKSpriteNode
- SKShapeNode
- SKEmitterNode
- SKEffectNode
- SKScene

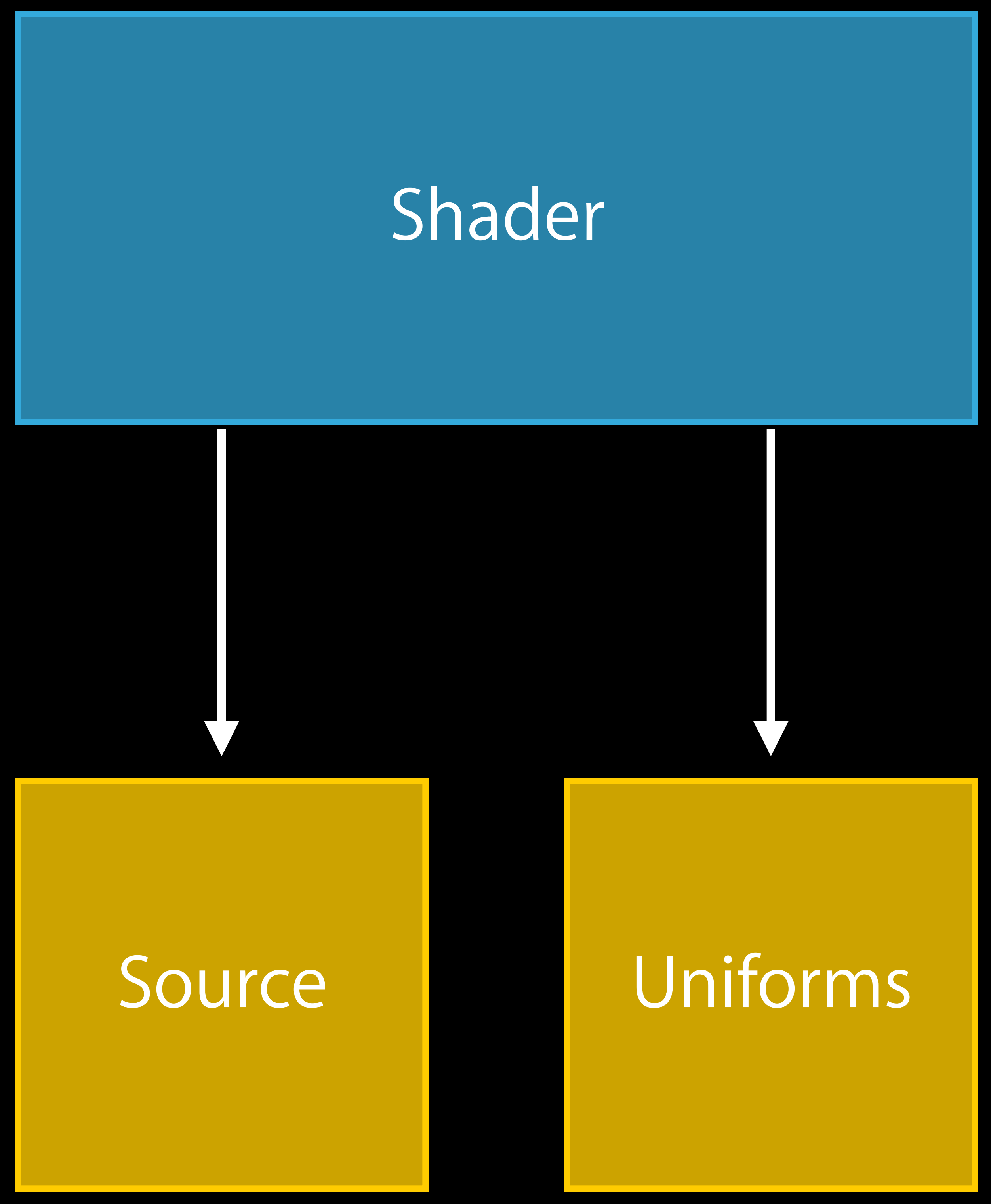
Built-in uniforms

- u_texture, v_tex_coord, u_sprite_size …

SKShader

Shader

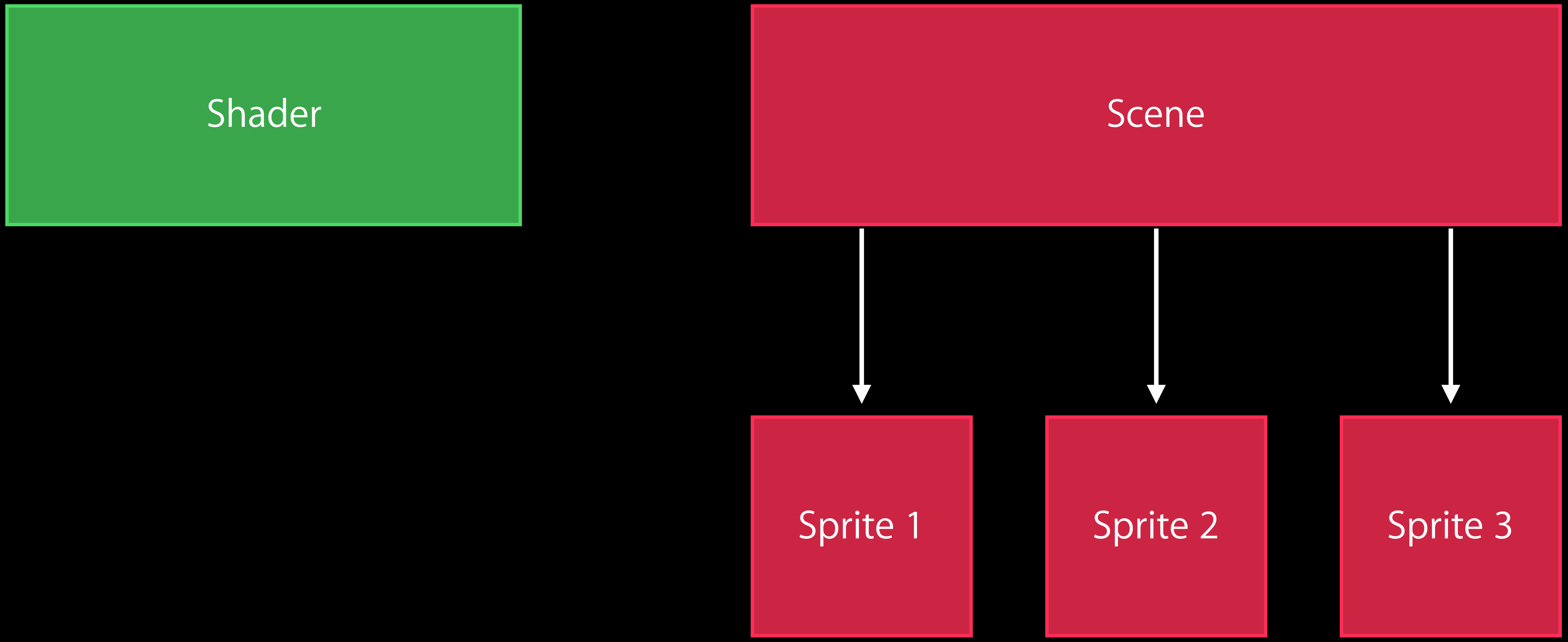
SKShader



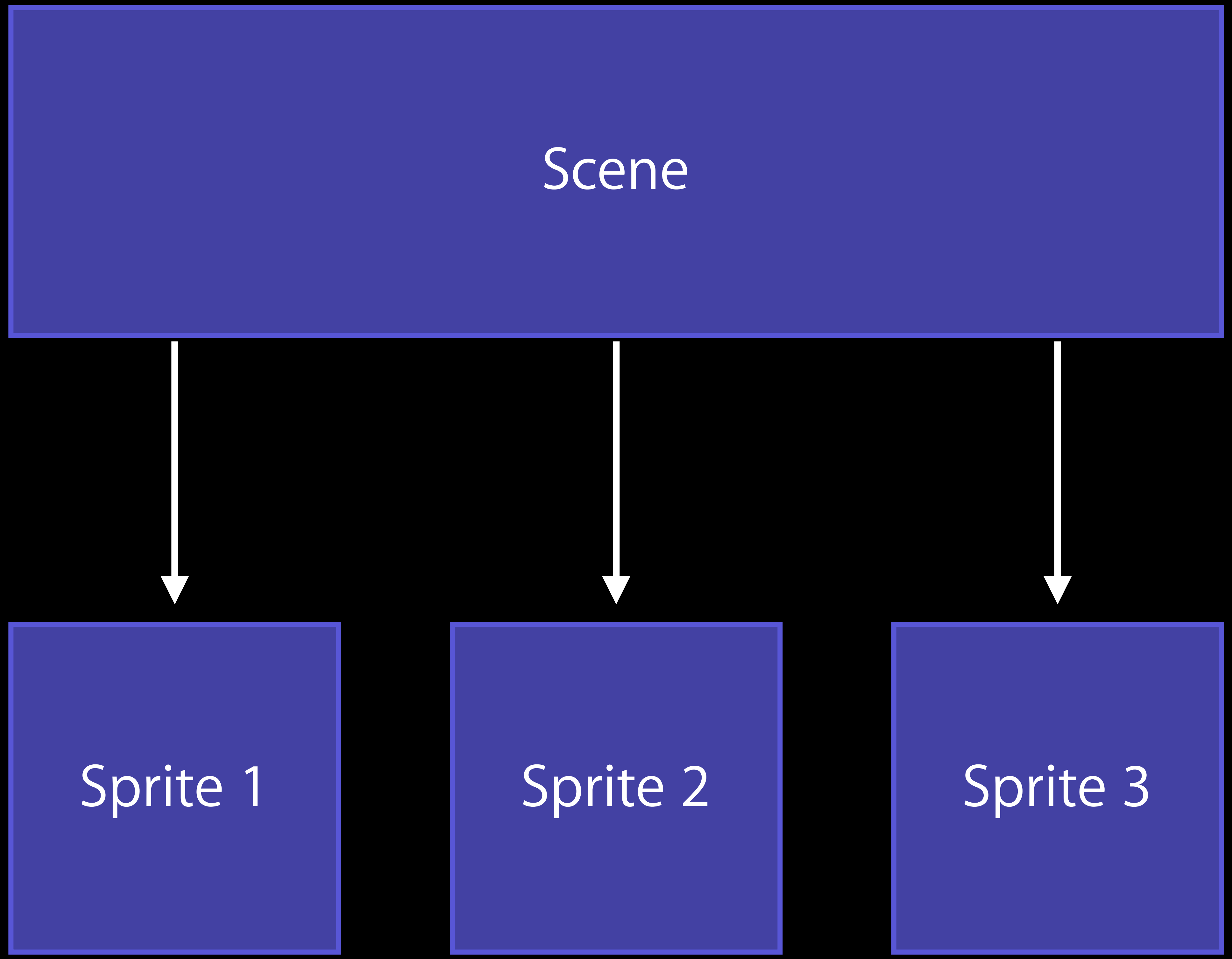
SKShader

Shader

SKShader



SKShader

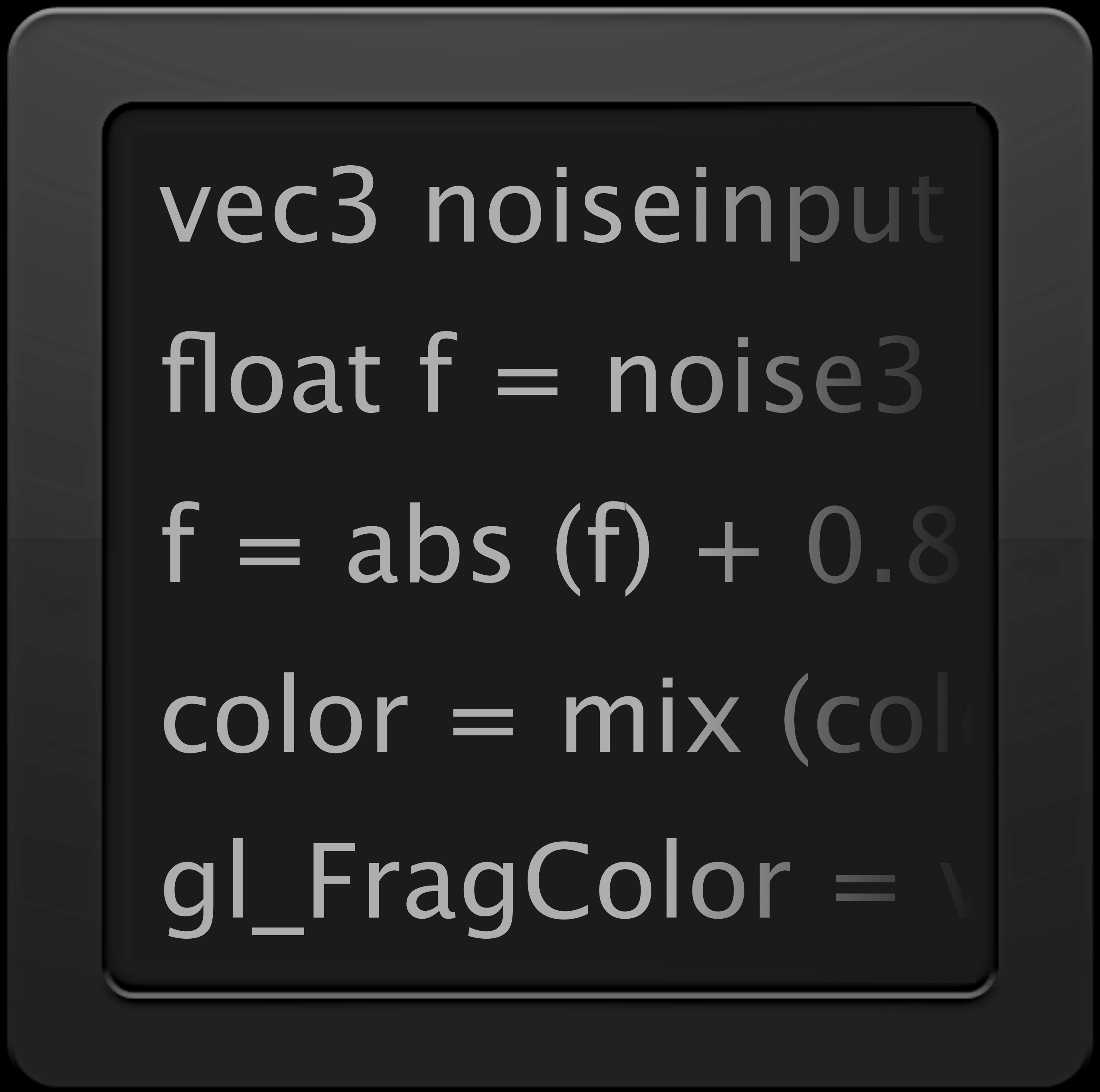


SKShader

Creation

Source code for a fragment shader

```
shader = [SKShader shaderWithFileNamed:@“blur.fsh"];
```



```
vec3 noiseinput
float f = noise3
f = abs (f) + 0.8
color = mix (col
gl_FragColor = v
```

Setting uniforms

```
shader.uniforms = @[[SKUniform uniformWithName:@"u_red" float:1],
                    [SKUniform uniformWithName:@"u_myTex" texture:nil]];
```

SKShader

Predefined shader symbols

Uniform

`u_texture`	`sampler2D`
`u_sprite_size`	`vec2`
`u_time`	`float`
`u_path_length`	`float`

Varying

`v_tex_coord`	`vec2`
`v_color_mix`	`vec4`
`v_path_distance`	`float`

Function

`SKDefaultShading`	`vec4`

Shaders

Best practices

Make use of built-in uniforms

- Avoid changing the shader's source
- Avoid adding/removing uniforms

Share shader objects whenever possible

- Initialize shader objects at load time
- Initialize shader using file than string

Shaders

Summary

Shaders allows custom rendering

Provides access to sprites properties

Add cool and unique effects

Shaders

Lighting and Shadows

New Physics

Integration with SceneKit

Tools

Improvements

Shaders

Lighting and Shadows

New Physics

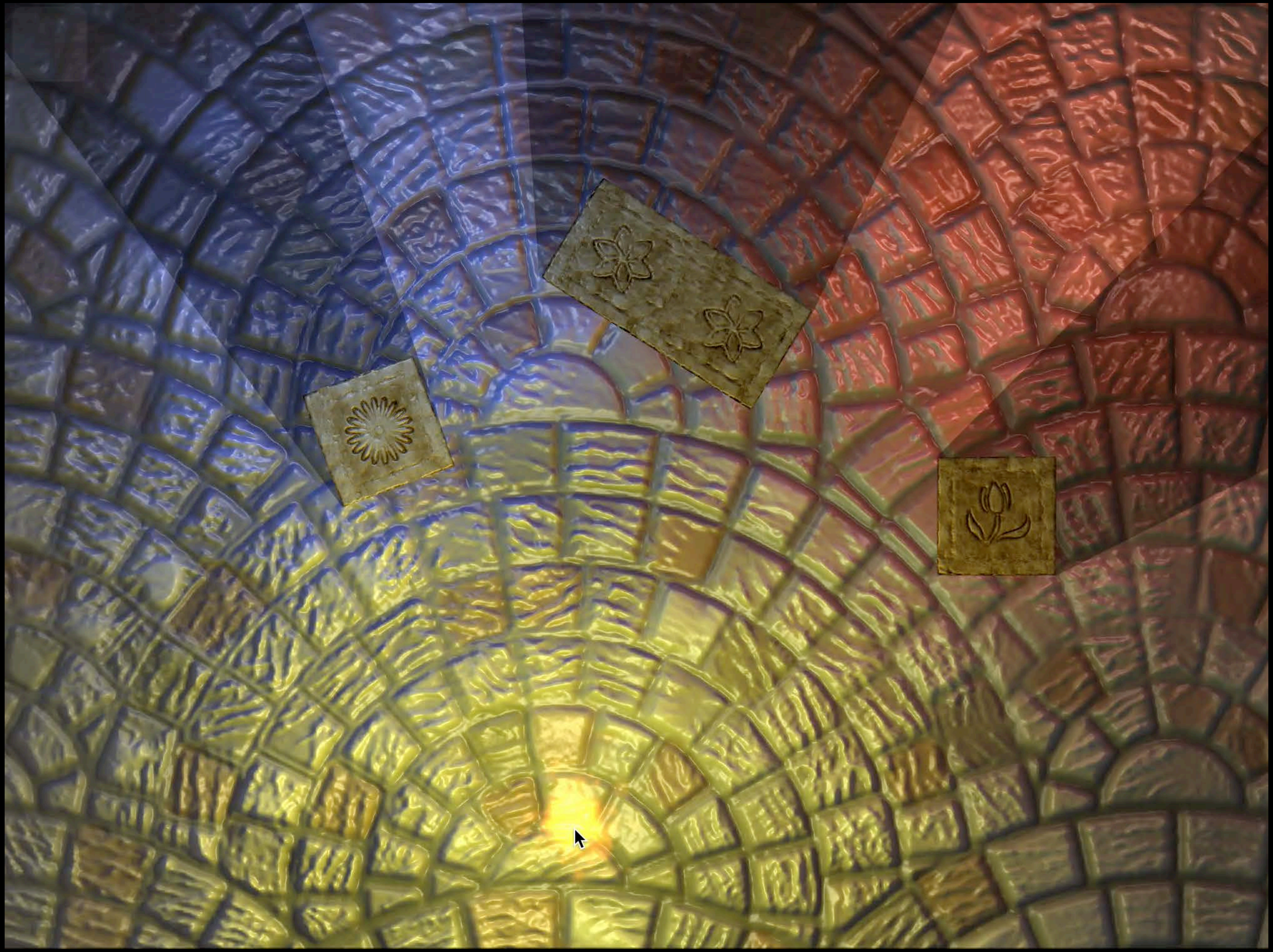
Integration with SceneKit

Tools

Improvements

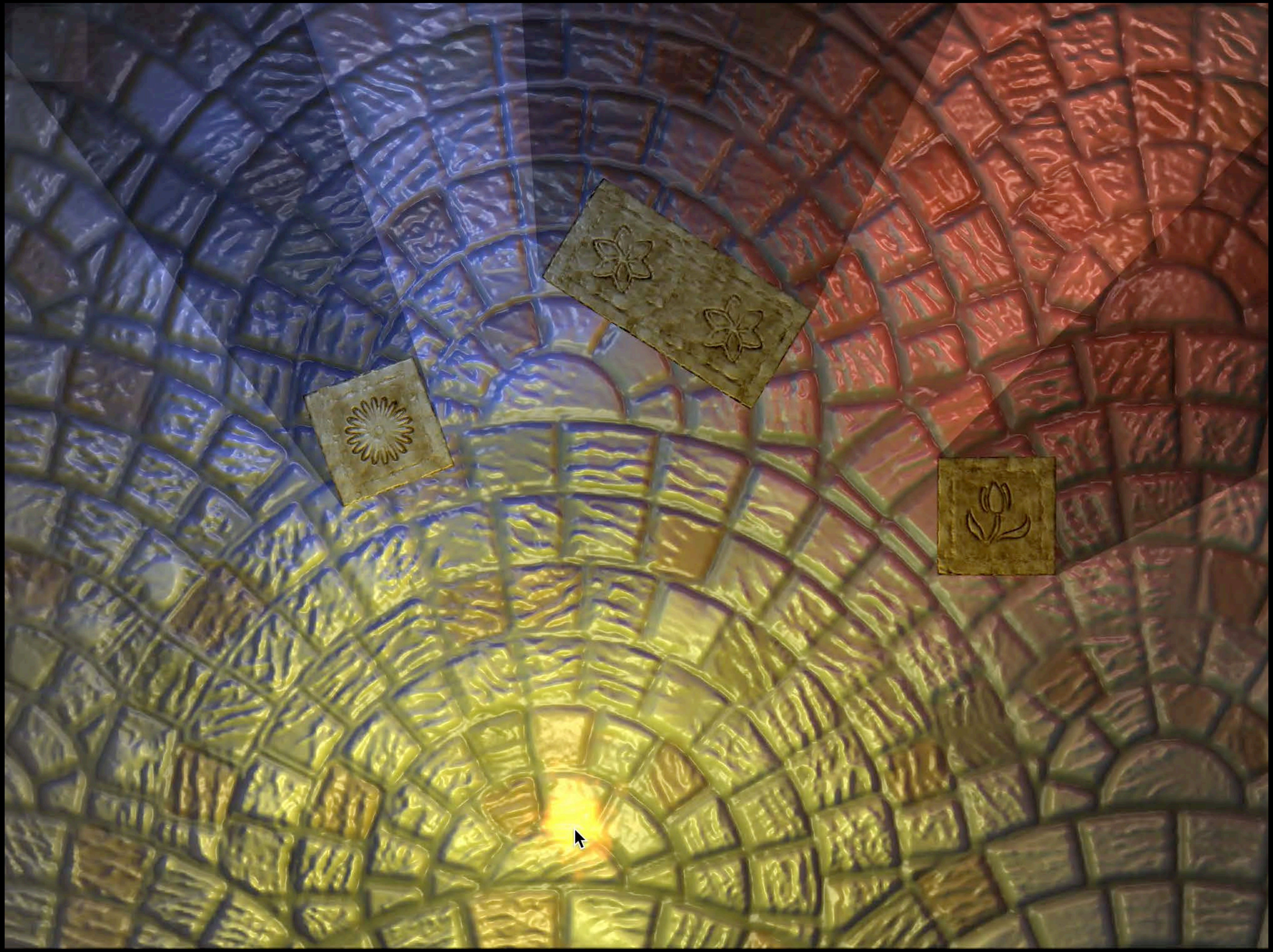
Lighting and Shadows

Introduction



Lighting and Shadows

Introduction



Lighting and Shadows

SKLightNode

- Position lights in your scene
- Light existing sprites
- Supports color, shadows, and falloff
- Up to eight different lights per sprite

Lighting and Shadows

SKLightNode

Position lights in your scene

Light existing sprites

Supports color, shadows, and falloff

Up to eight different lights per sprite

Just another SKNode

Lighting and Shadows

SKLightNode

Lighting and Shadows

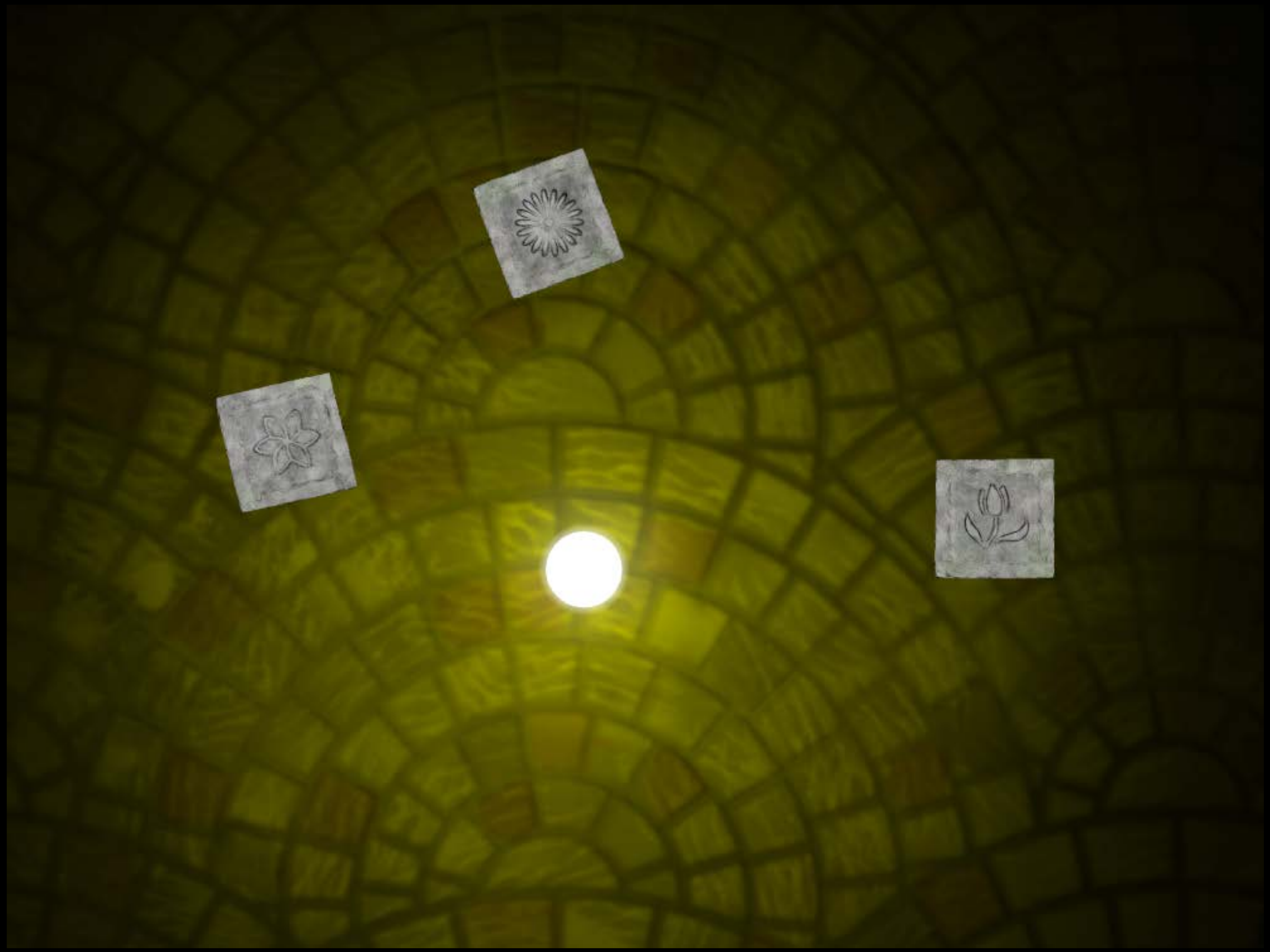
SKLightNode

lightColor 🟨

Lighting and Shadows

SKLightNode

lightColor 🟨

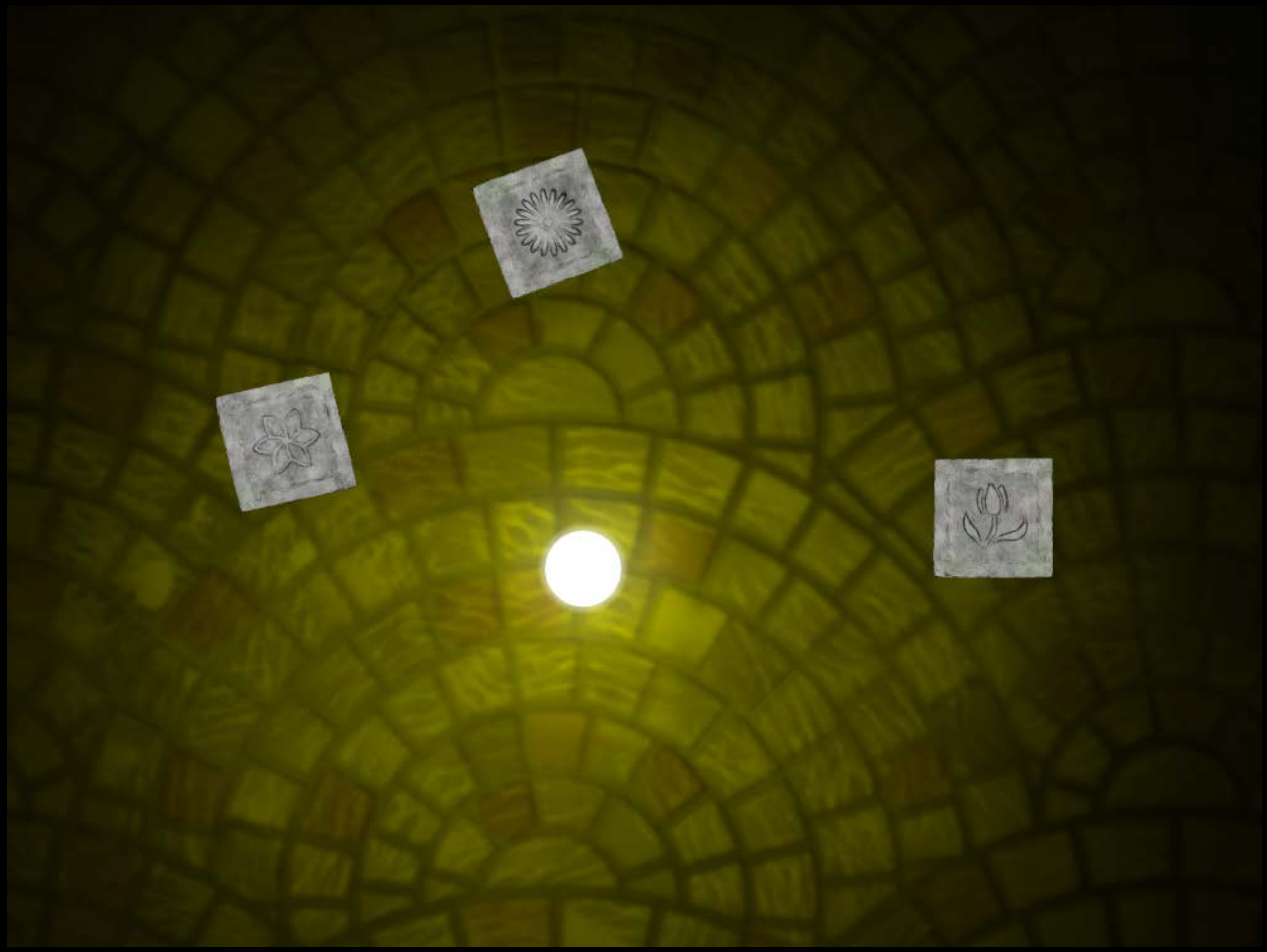


Lighting and Shadows

SKLightNode

lightColor 🟨

shadowColor ◼

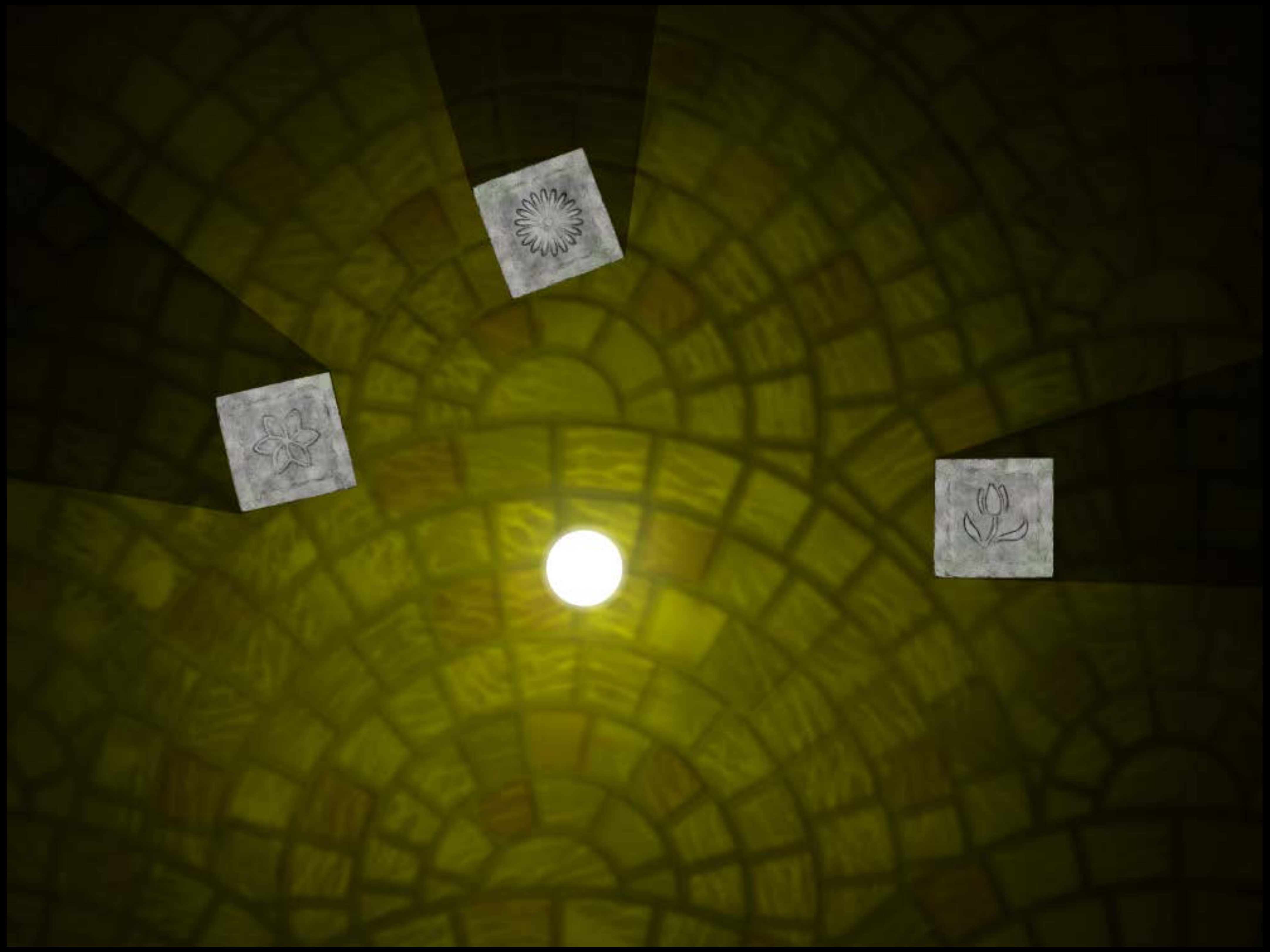


Lighting and Shadows

SKLightNode

lightColor 🟨

shadowColor ⬛



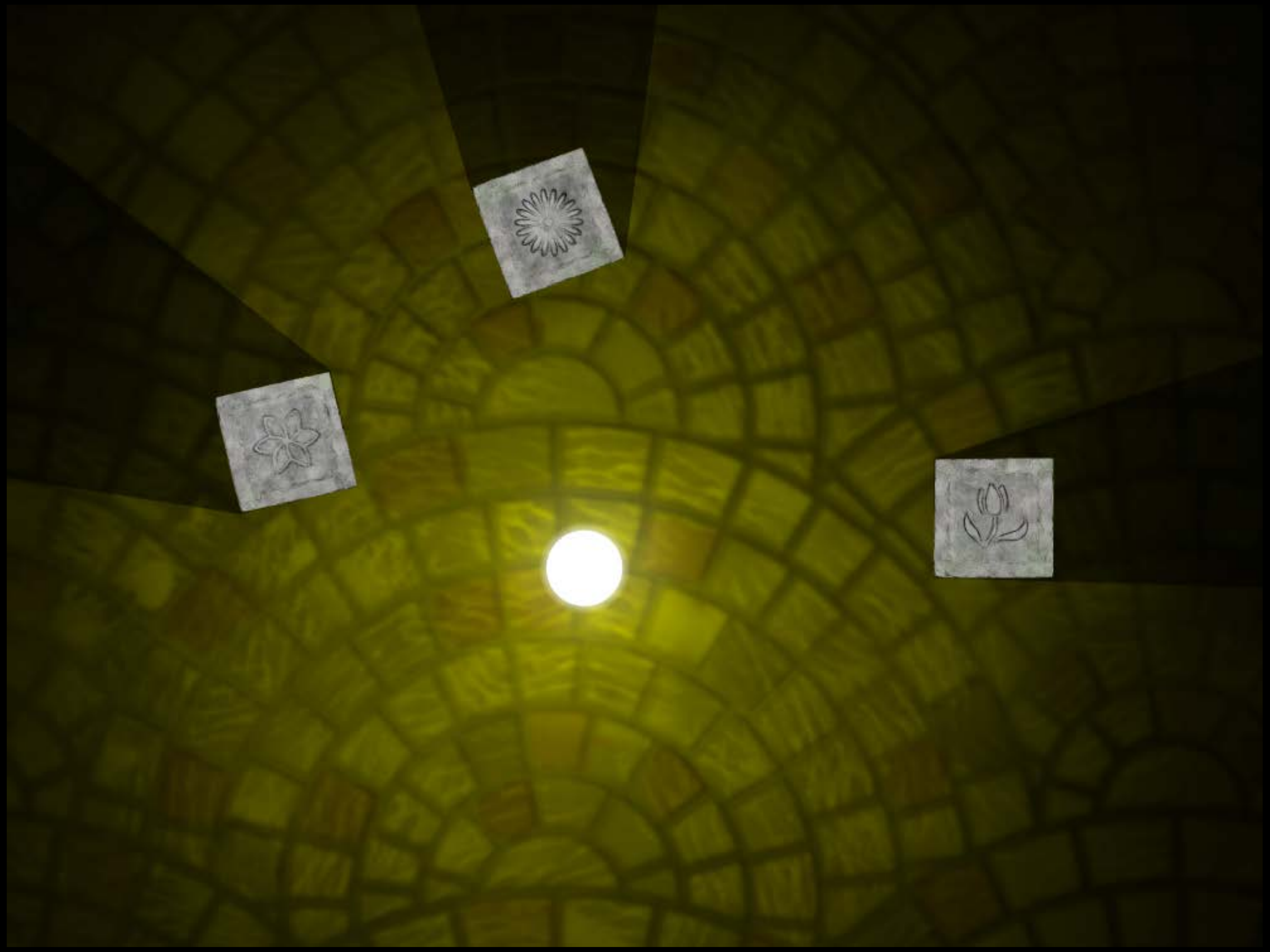
Lighting and Shadows

SKLightNode

lightColor 🟨

shadowColor ■

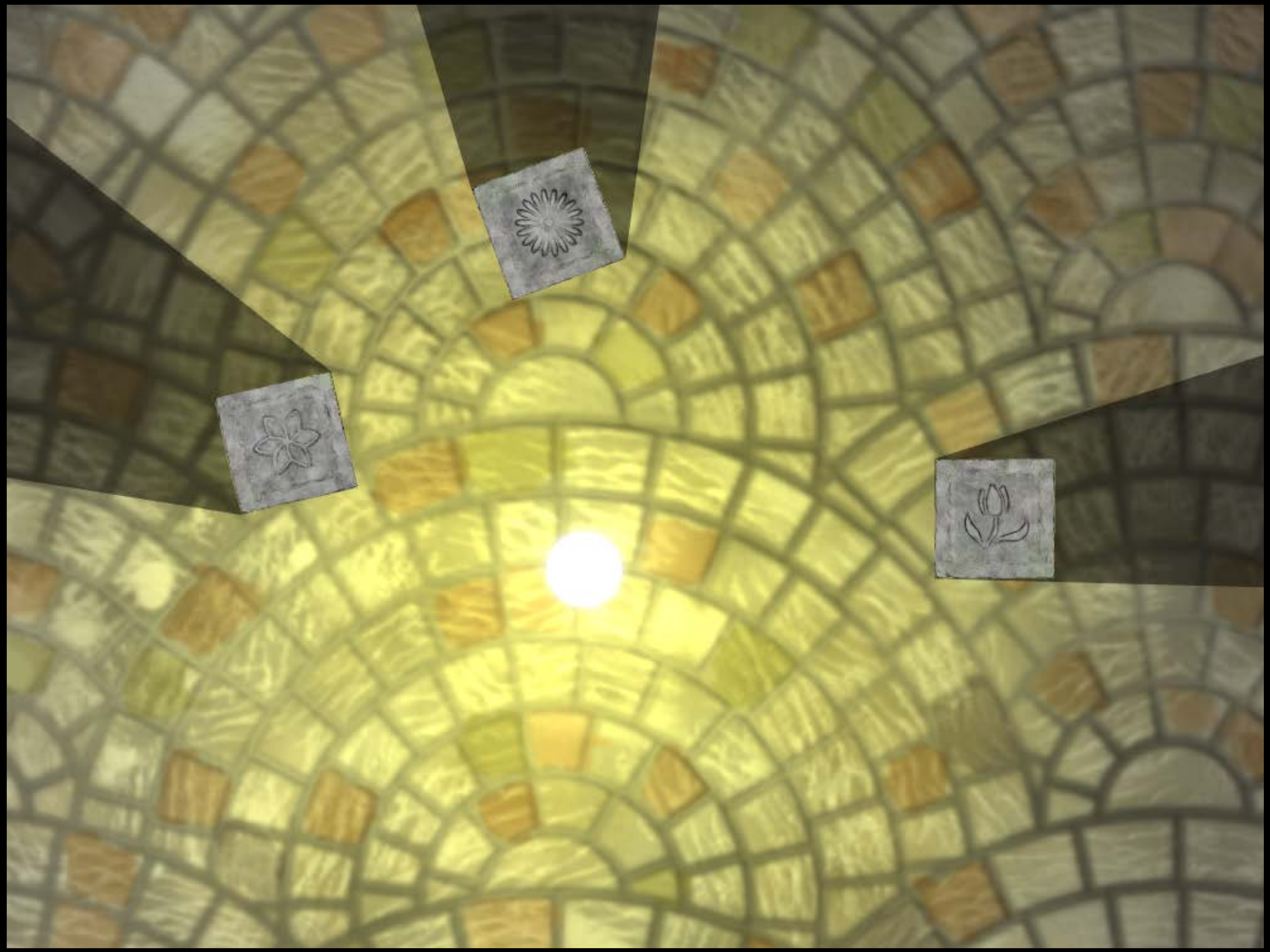
ambientColor □



Lighting and Shadows

SKLightNode

lightColor	🟨
shadowColor	◼
ambientColor	⬜



Lighting and Shadows

SKLightNode

falloff

- Does not effect ambient

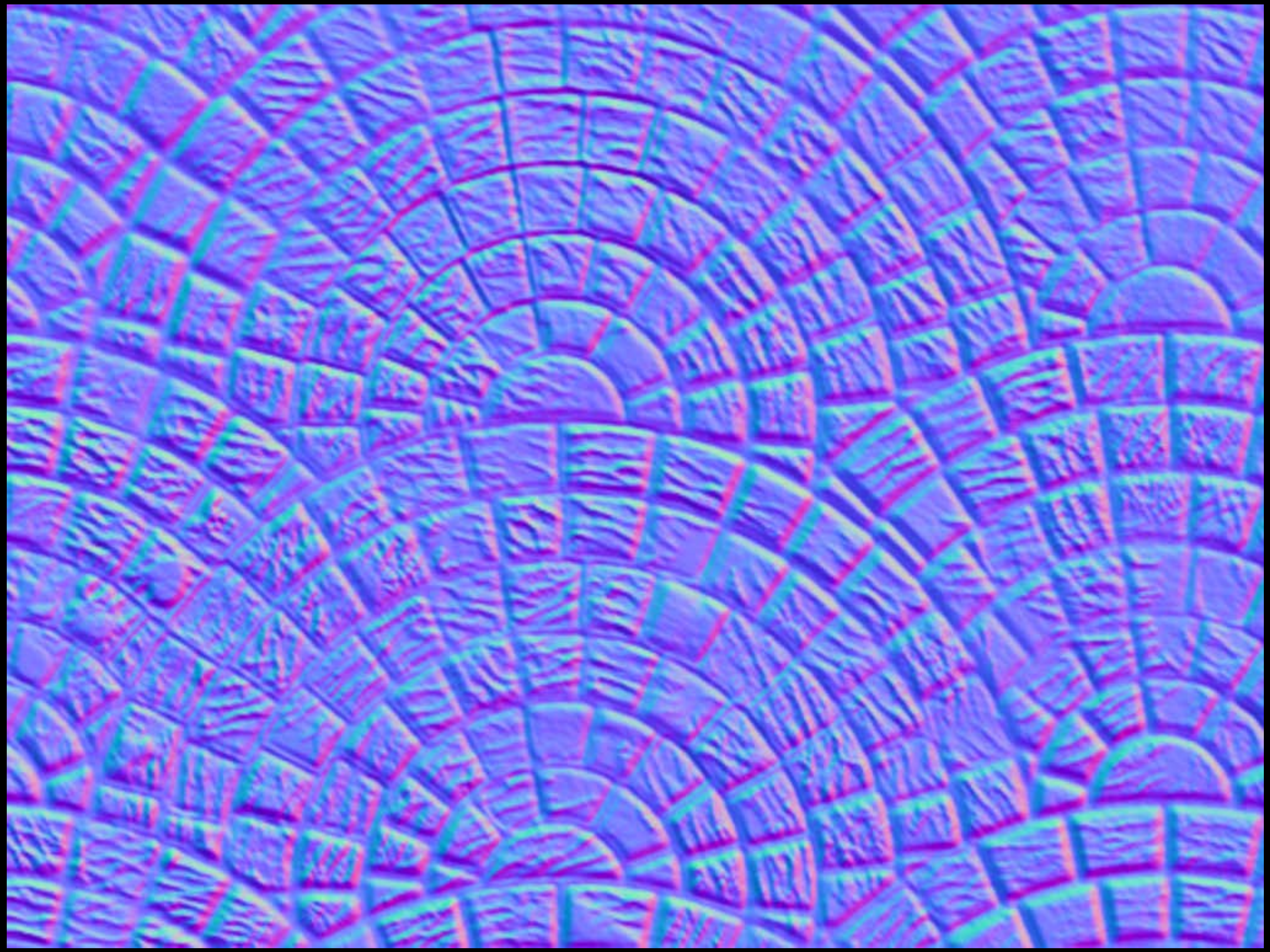
categoryBitMask

- `SKSpriteNode.lightingBitMask`
- `SKSpriteNode.shadowCastBitMask`
- `SKSpriteNode.shadowedBitMask`

Lighting and Shadows

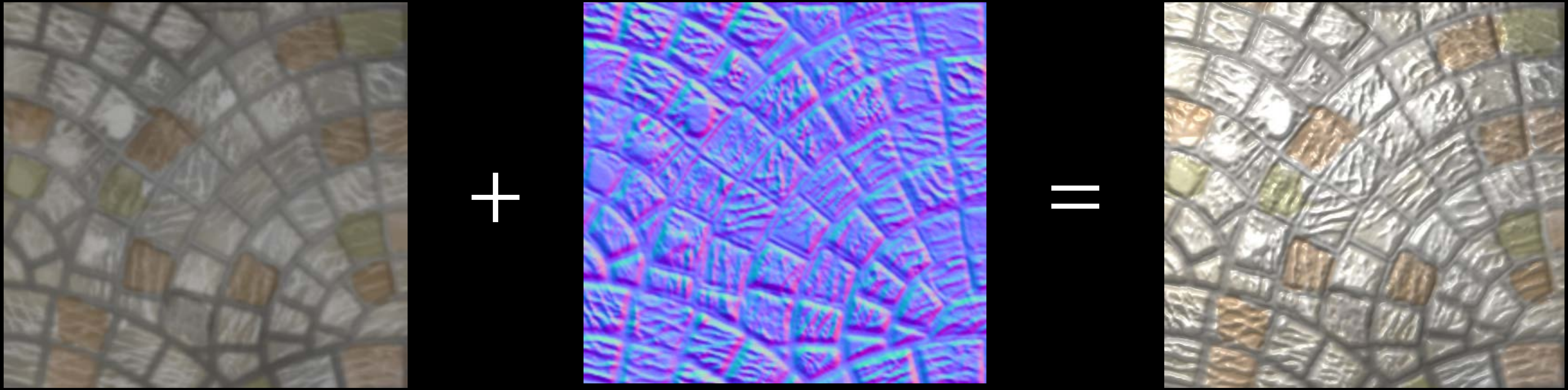
SKSpriteNode

`normalTexture`



Lighting and Shadows

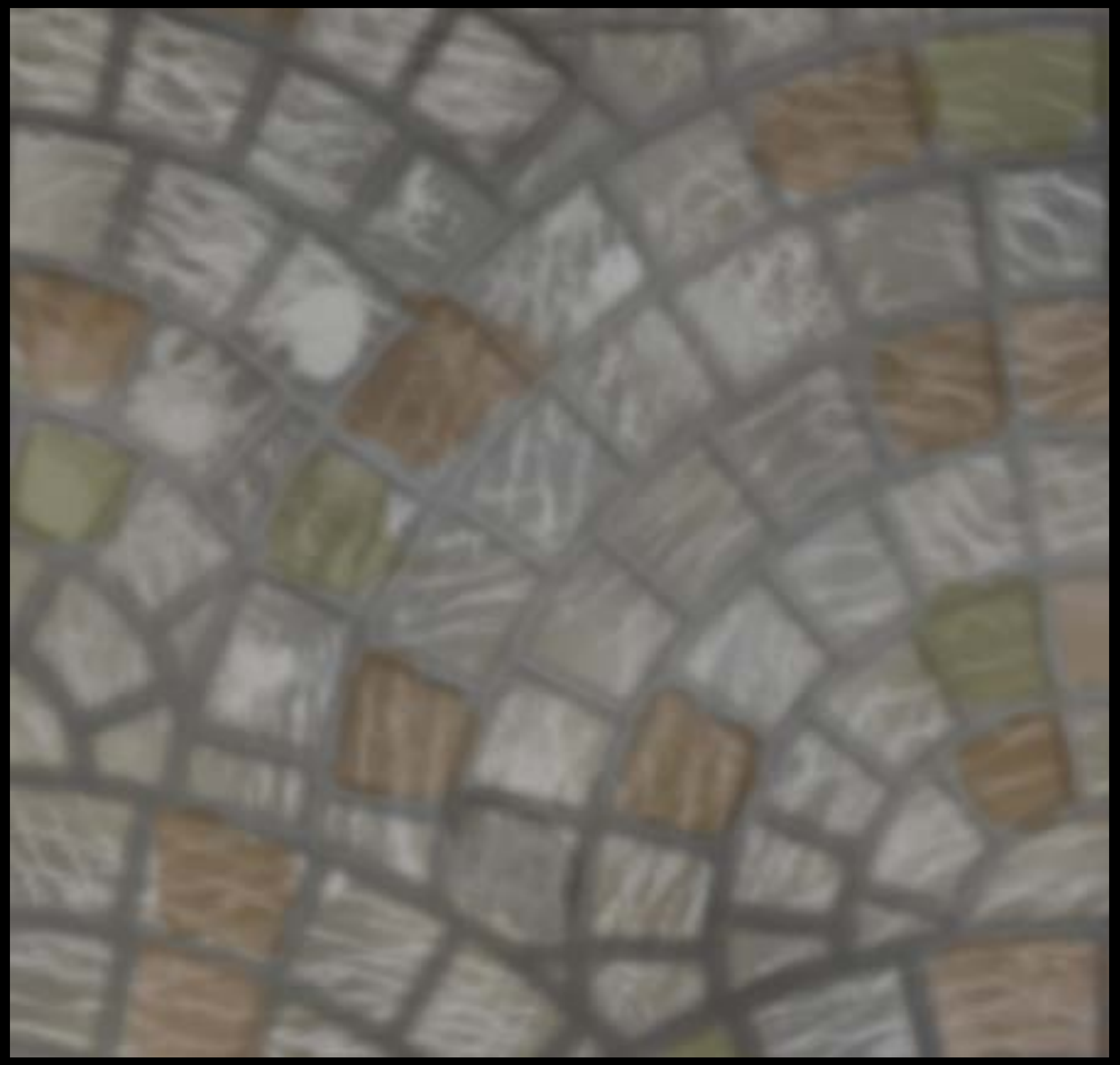
Normal map



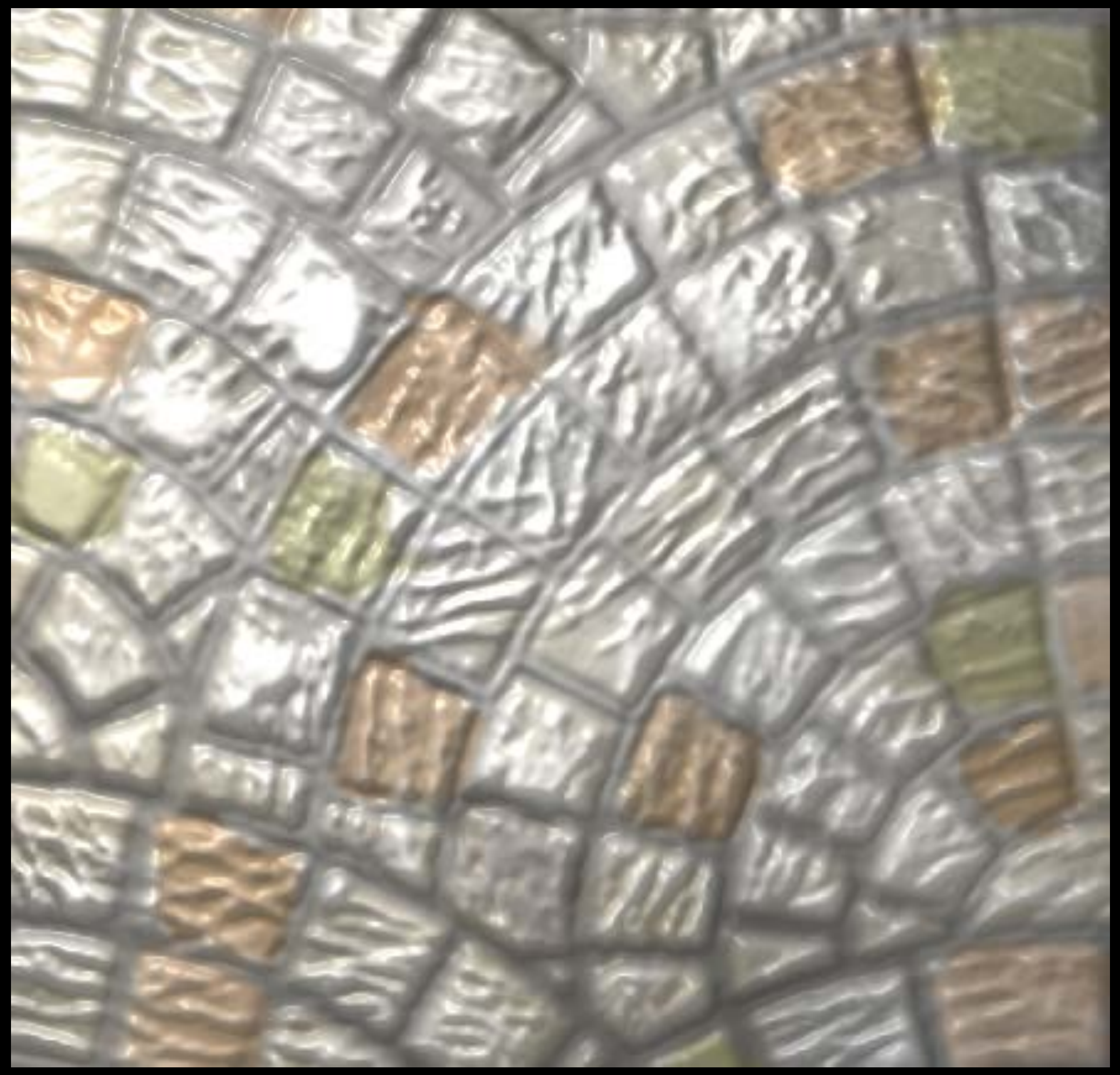
```
sprite.normalTexture = [SKTexture textureWithImageNamed:@“normal”];
```

Lighting and Shadows

Automatic normal map



=



```
sprite.normalTexture = [myTex textureByGeneratingNormalMap];
```

Lighting and Shadows

Automatic normal map

Tuning automatic normal map

- Smoothness
- Contrast

```
-(instancetype)textureByGeneratingNormalMapWithSmoothness:(CGFloat)smoothness
                                                 contrast:(CGFloat)contrast
```

Lighting and Shadows

Automatic normal map

Tuning automatic normal map

- Smoothness
- Contrast

```
-(instancetype)textureByGeneratingNormalMapWithSmoothness:(CGFloat)smoothness
                                                 contrast:(CGFloat)contrast
```

Lighting and Shadows

Summary

Lights are very easy to use

Automatic normal map provides dynamic look

Performance best practice

- Number of lights on the same sprite

Shaders

Lighting and Shadows

New Physics

Integration with SceneKit

Tools

Improvements

Shaders

Lighting and Shadows

New Physics

Integration with SceneKit

Tools

Improvements

New Physics

Per-pixel physics

Constraints

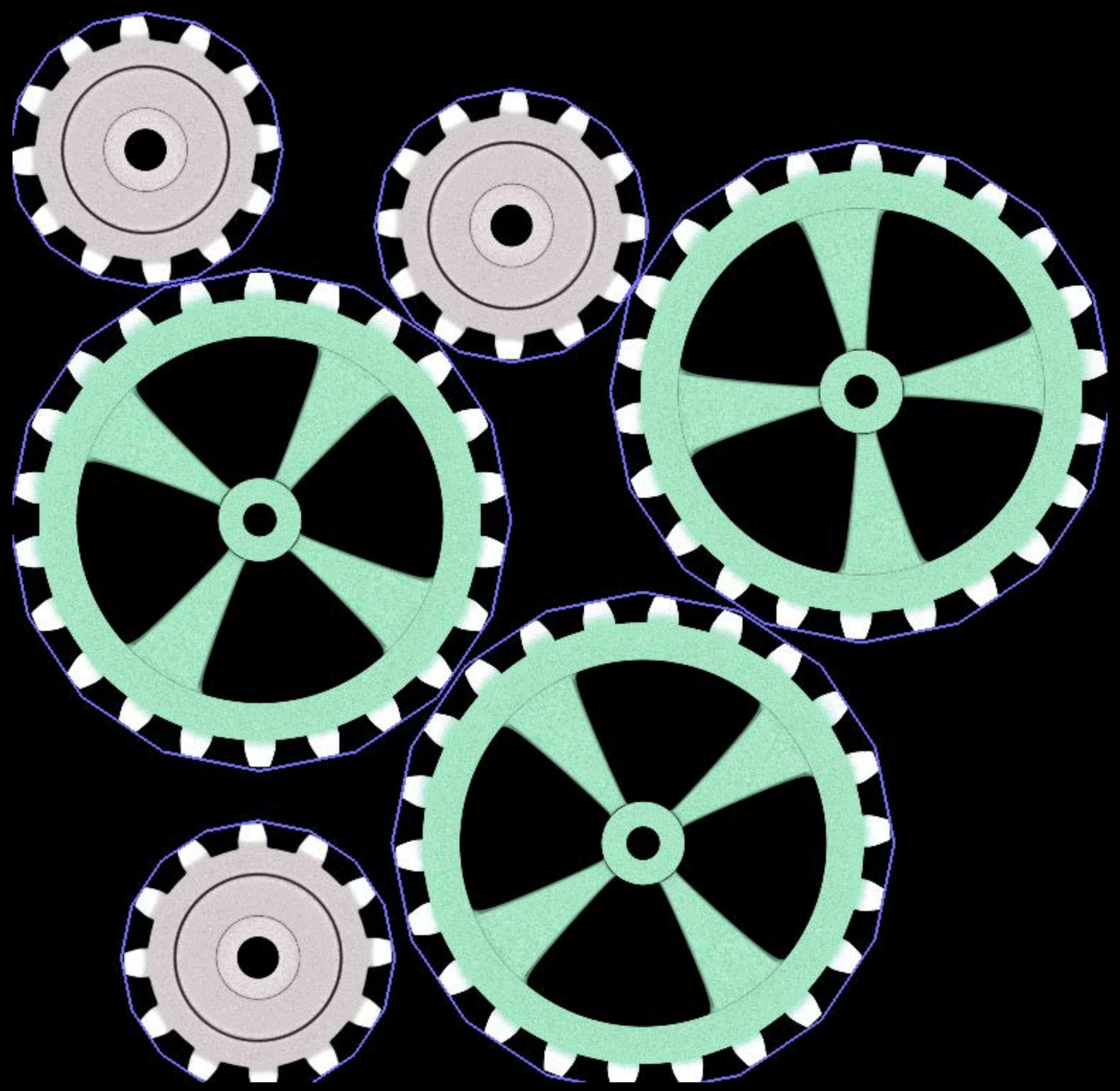
Inverse kinematics

Physics fields

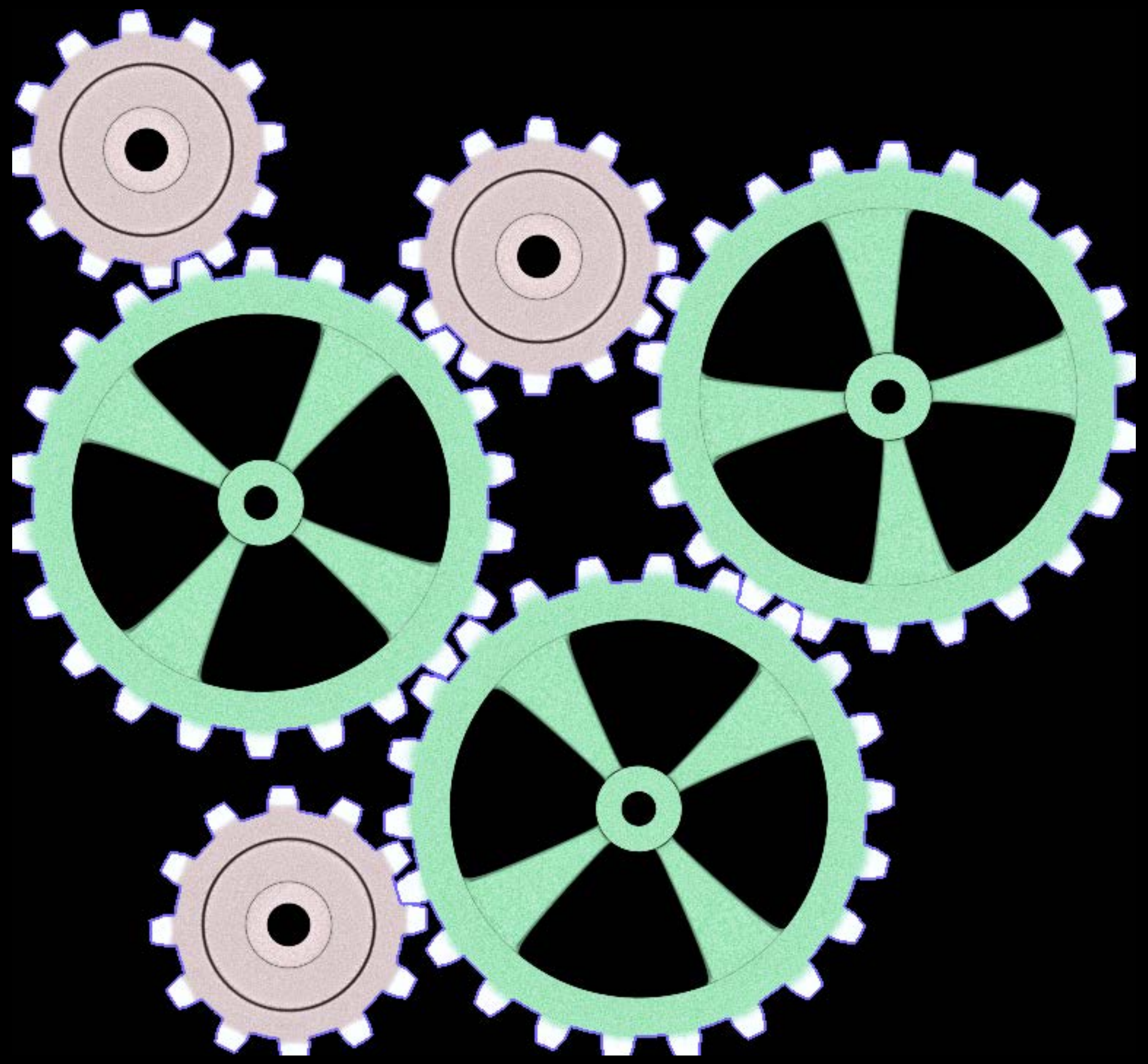
Per-Pixel Physics

Per-Pixel Physics

Introduction



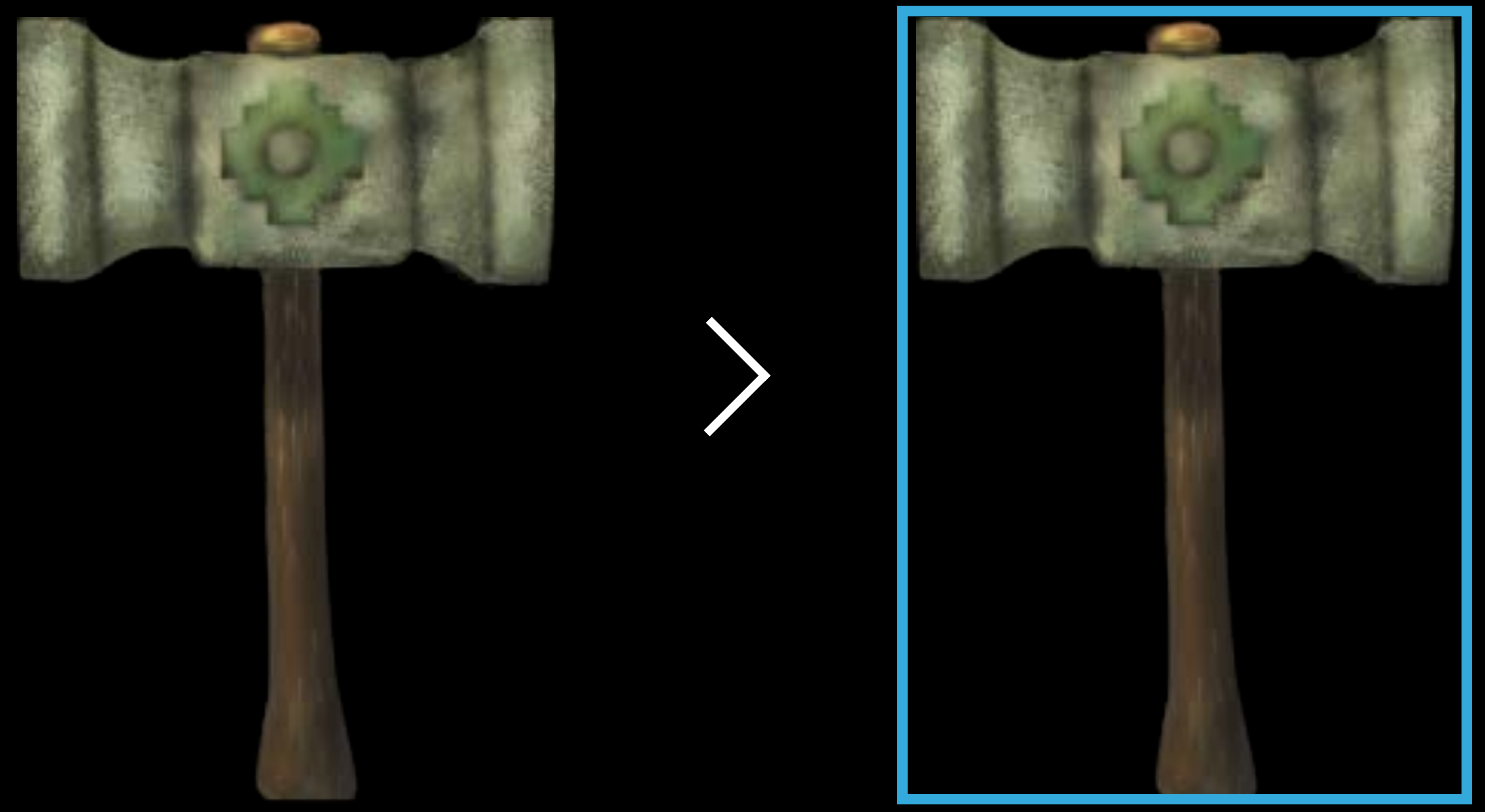
>



SKPhysicsBody

Initialization

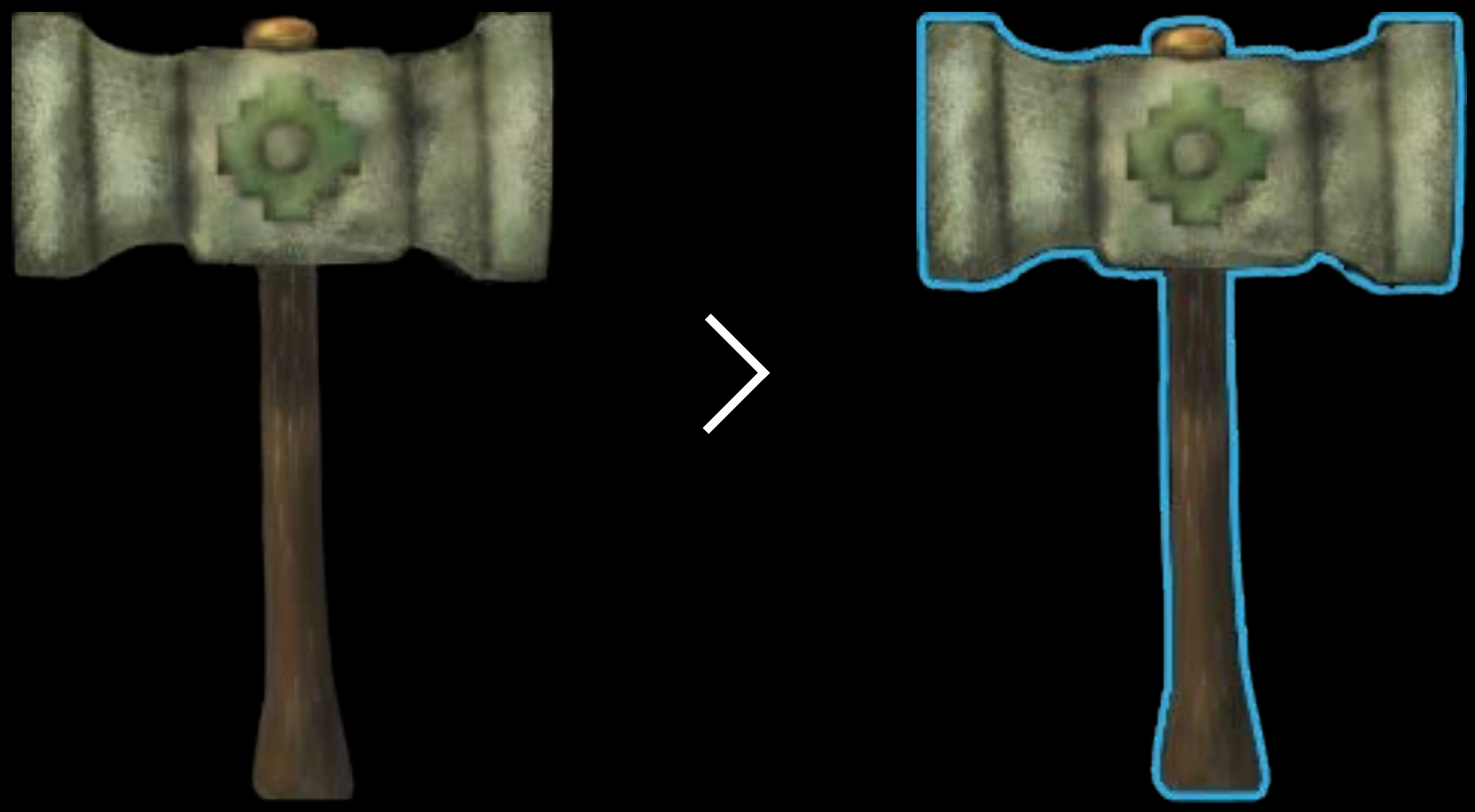
```
hammer.physicsBody = [SKPhysicsBody bodyWithRectangleOfSize:hammer.size];
```



SKPhysicsBody

Per-pixel physics initialization

```
hammer.physicsBody = [SKPhysicsBody bodyWithTexture:hammer.texture
                                                   size:hammer.size];
```



SKPhysicsBody

Initialization

```
+ (SKPhysicsBody *)bodyWithTexture:(SKTexture *)texture
                   alphaThreshold:(float)alphaThreshold
                             size:(CGSize)size
```

alphaThreshold the alpha value above which a pixel is interpreted as opaque

Per-Pixel Physics

Summary

Easy creation

Accurate representation

Performance

- Provides a good balance between performance and accuracy
- Texture size matters
- Limit the number of per-pixel physics bodies

Constraints

Constraints

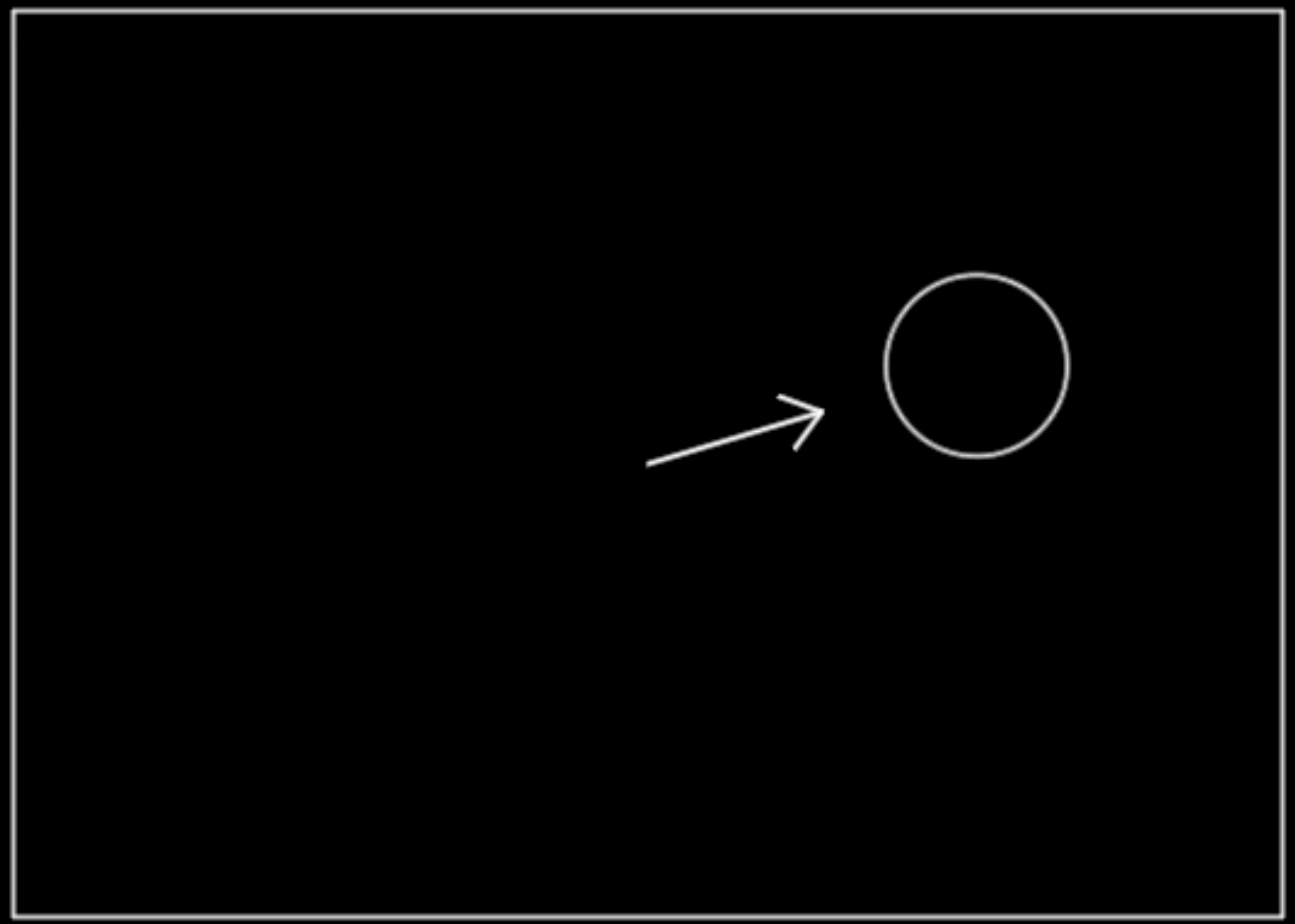
Introduction

Remove boilerplate logic in game code

Applied after physics update

Interactions of constraints

- Cannon
- Runway
- Health indication



Constraints

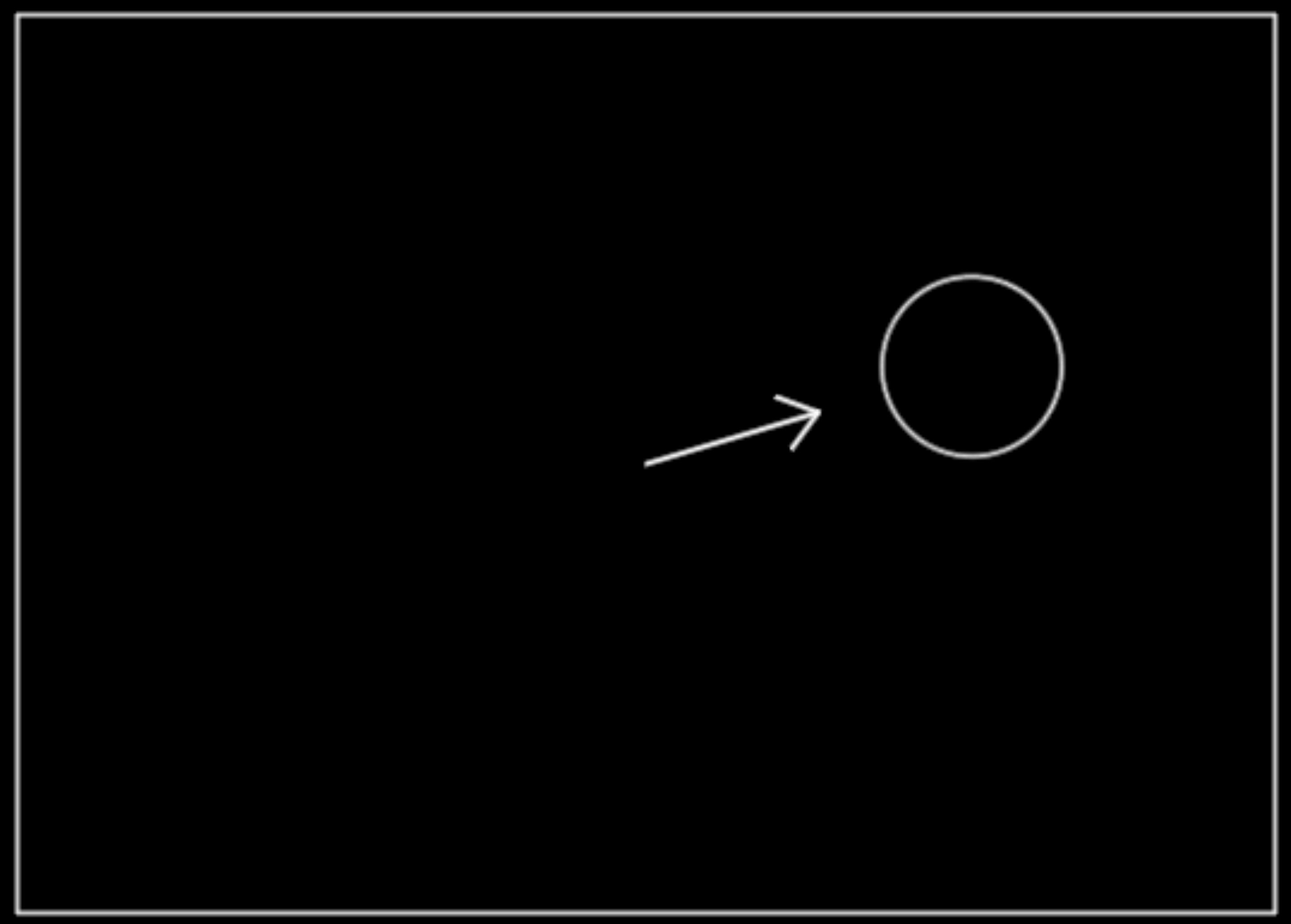
Introduction

Remove boilerplate logic in game code

Applied after physics update

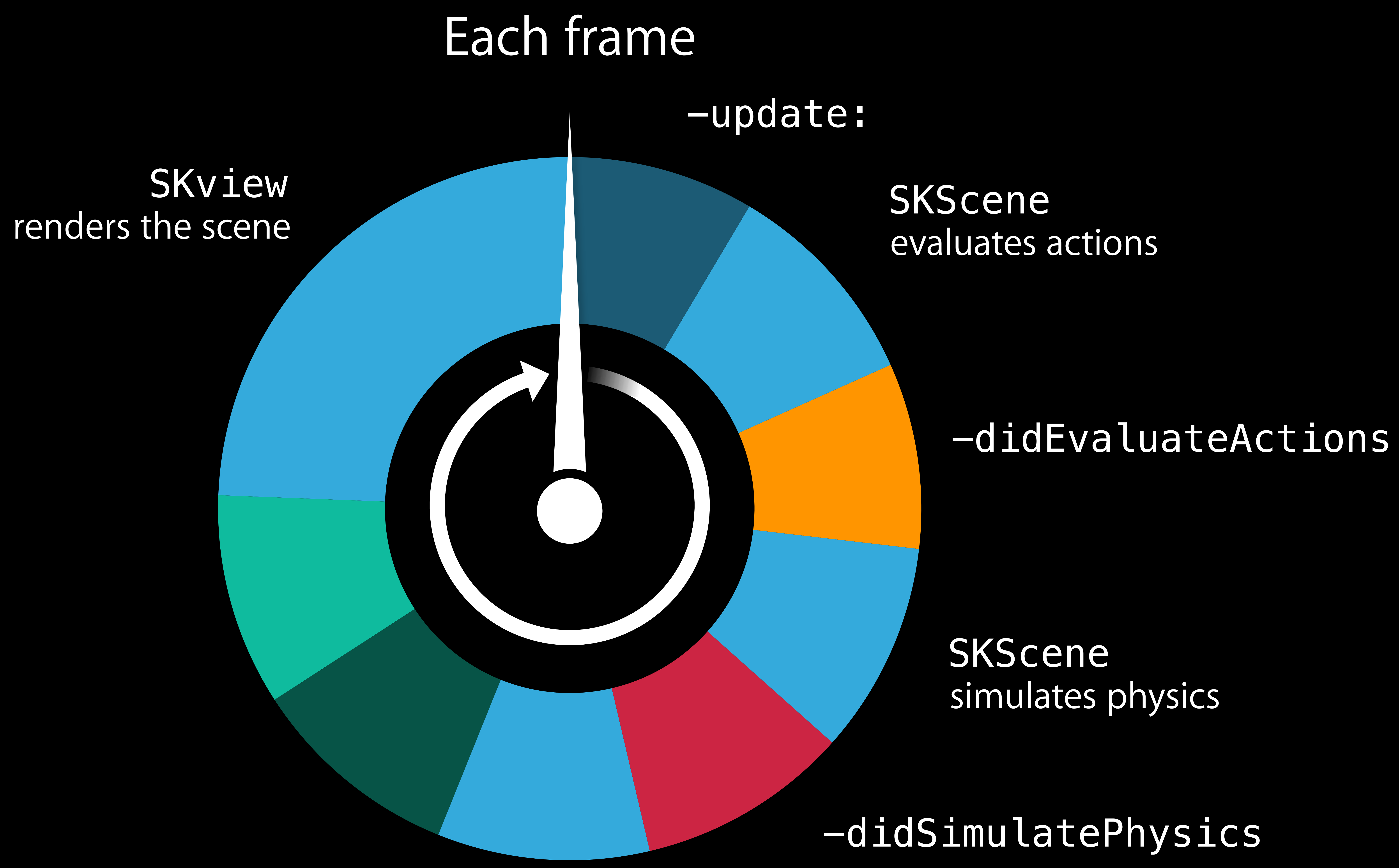
Interactions of constraints

- Cannon
- Runway
- Health indication



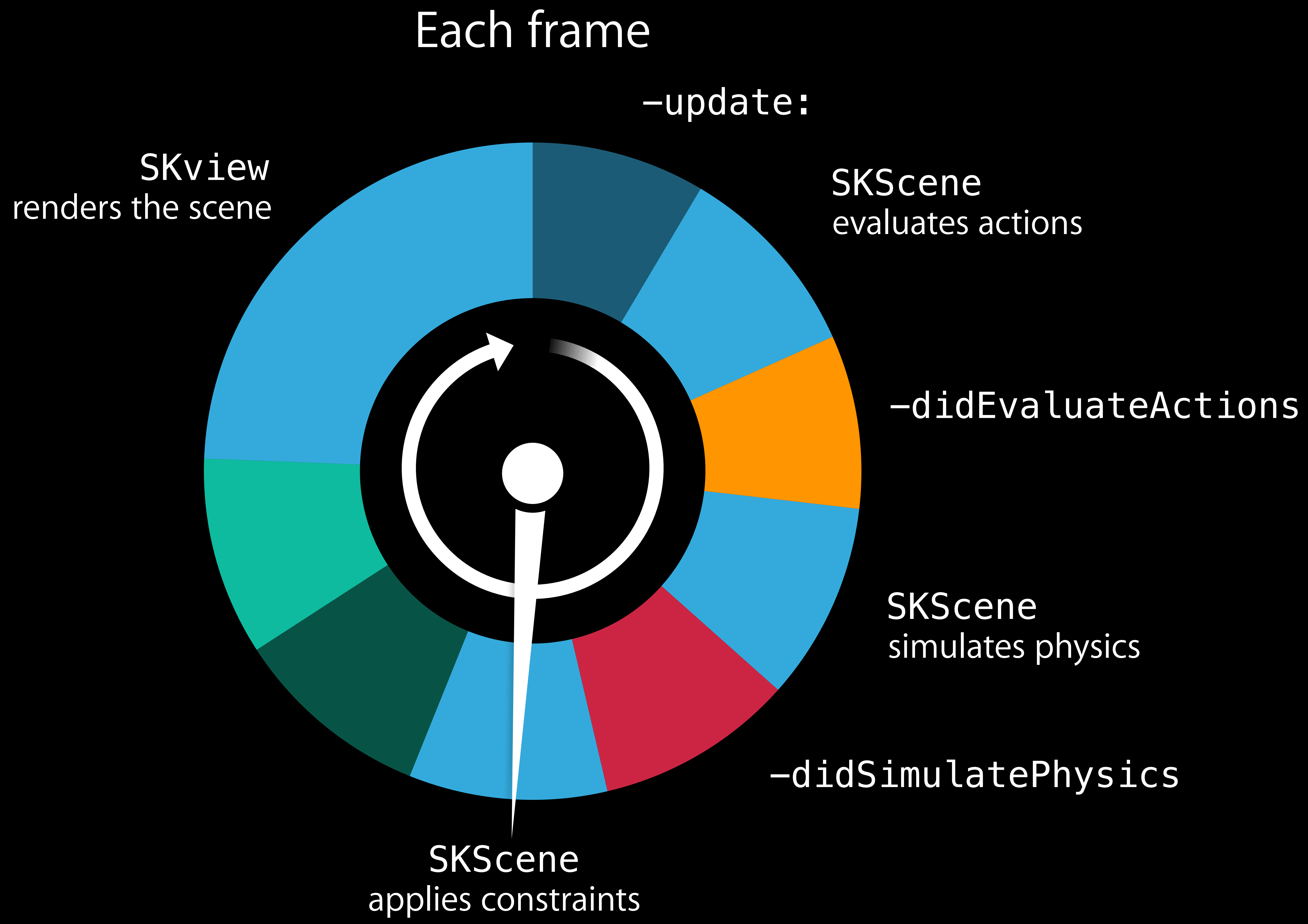
Constraints

In the loop



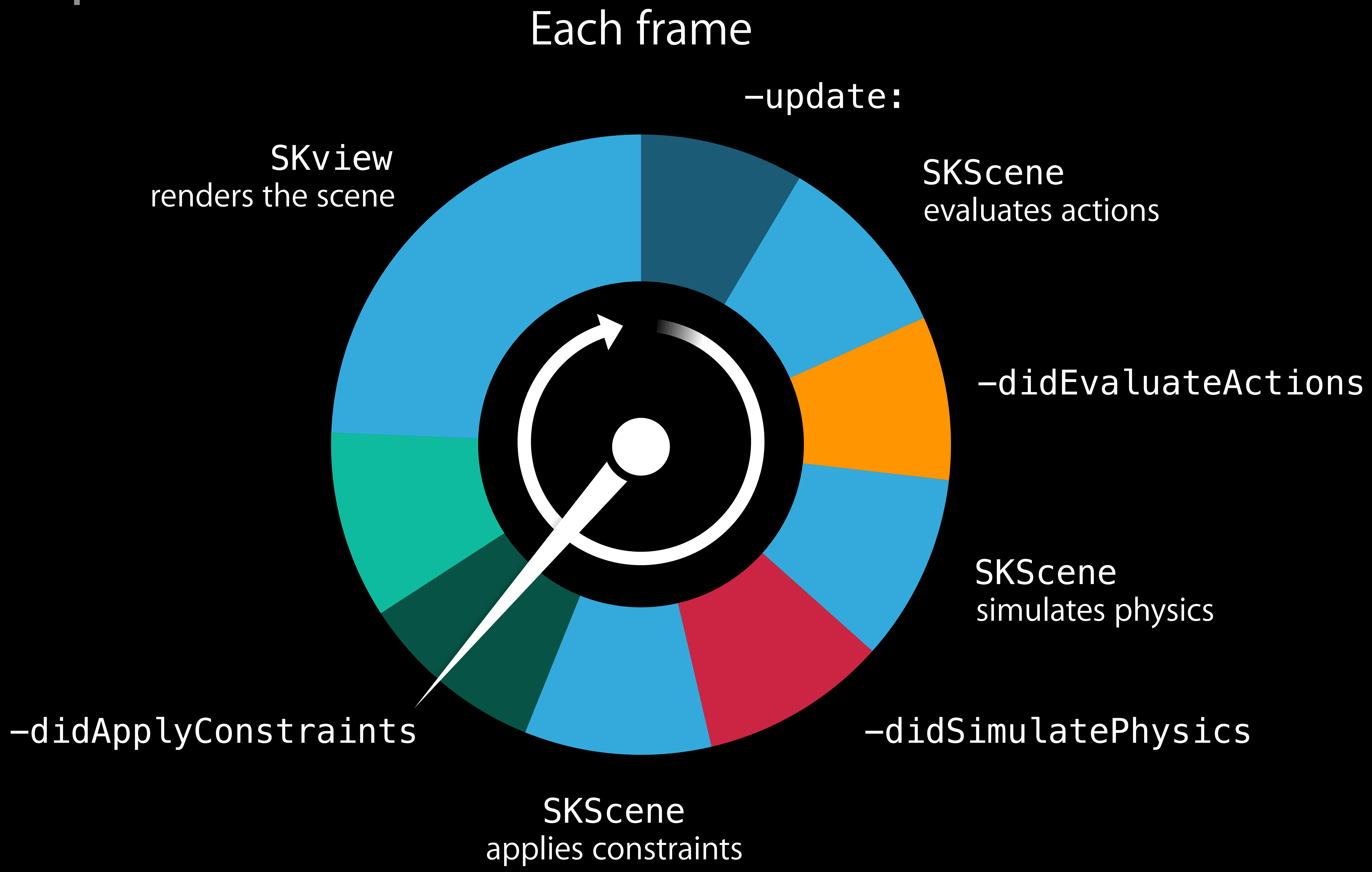
Constraints

In the loop



Constraints

In the loop



Constraints

Basics

New object—SKConstraint

Defines a mathematical constraint on one property of a node

Constraints are attached to nodes

Scene applies constraints attached to nodes

Constraints

Properties

Position

Orientation

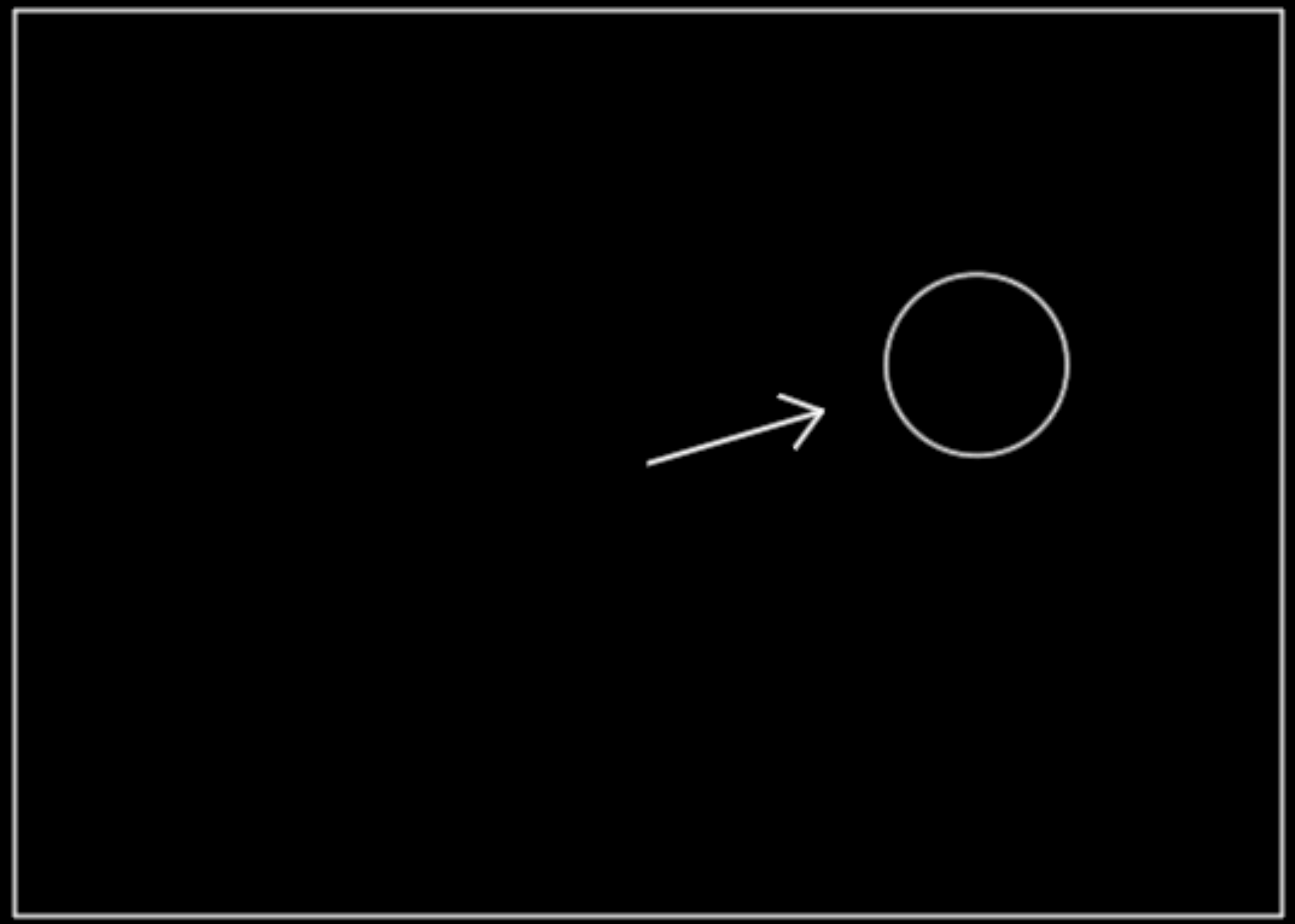
Distance

Enable/Disable

Example

Orient to node

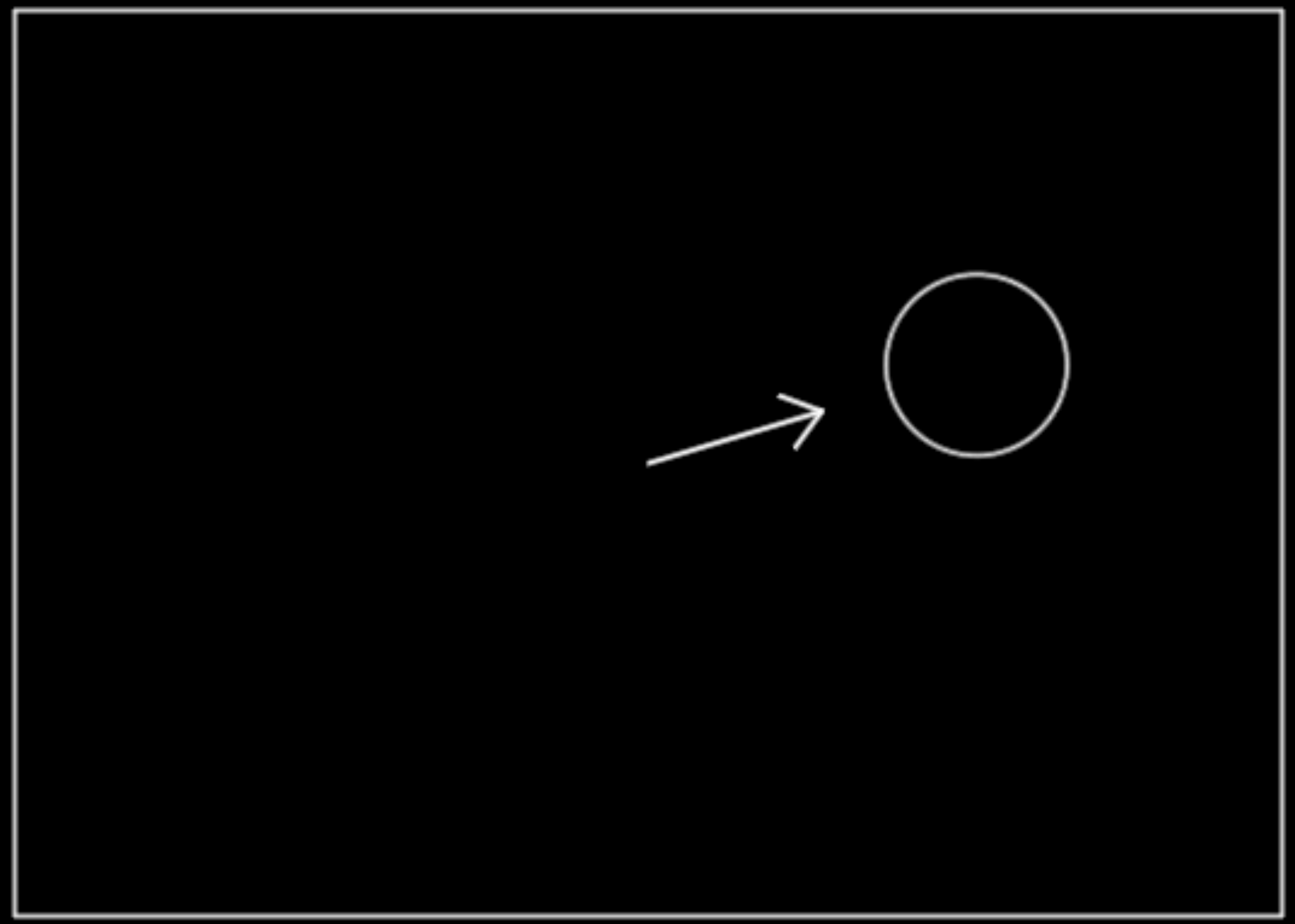
```
SKRange* range = [SKRange rangeWithConstantValue:0.0f];

SKConstraint* orientConstraint = [SKConstraint orientToNode:targetNode
                                                      offset:range];

node.constraints = @[orientConstraint];
```



Example

Orient to node

```
SKRange* range = [SKRange rangeWithConstantValue:0.0f];

SKConstraint* orientConstraint = [SKConstraint orientToNode:targetNode
                                                     offset:range];

node.constraints = @[orientConstraint];
```



Example

Position constraint

```
SKRange* range = [SKRange rangeWithLowerLimit:-100.0f upperLimit:100.0f]];
```

Example

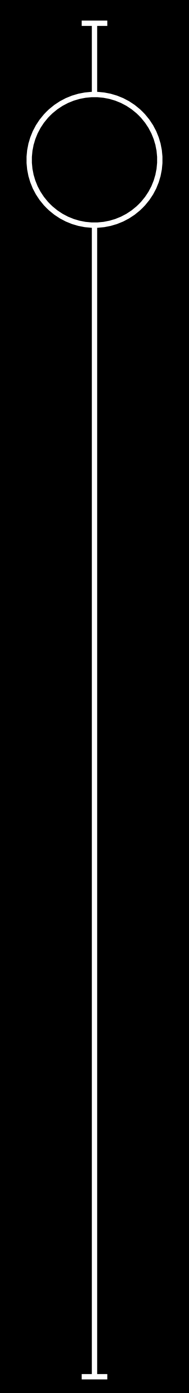
Position constraint

```
SKRange* range = [SKRange rangeWithLowerLimit:-100.0f upperLimit:100.0f]];

SKConstraint* constraint1 = [SKConstraint positionX:range]; //X constraint
```

Example

Position constraint

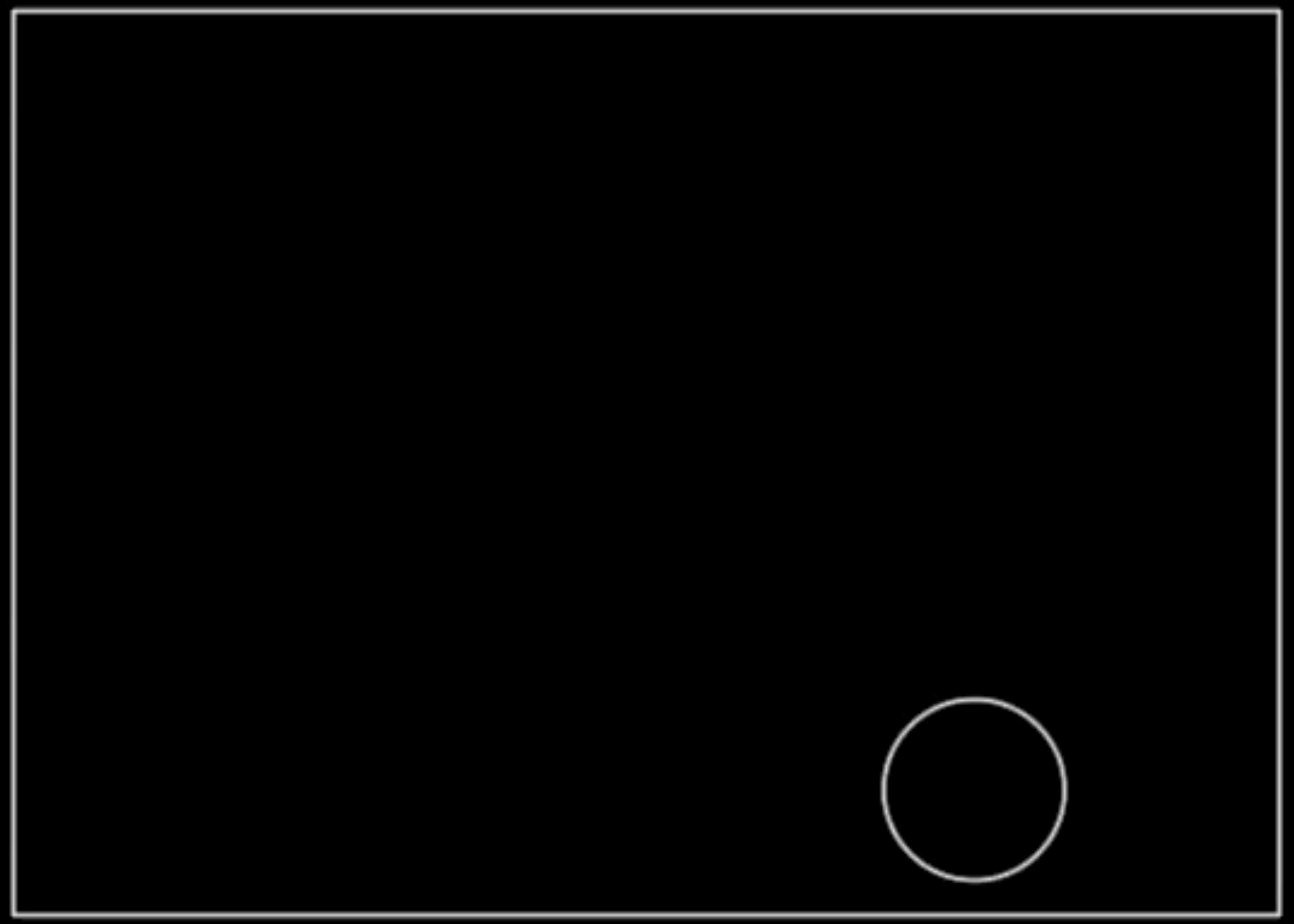
```
SKRange* range = [SKRange rangeWithLowerLimit:-100.0f upperLimit:100.0f]];

SKConstraint* constraint2 = [SKConstraint positionY:range]; //Y constraint
```



Example

Position constraint

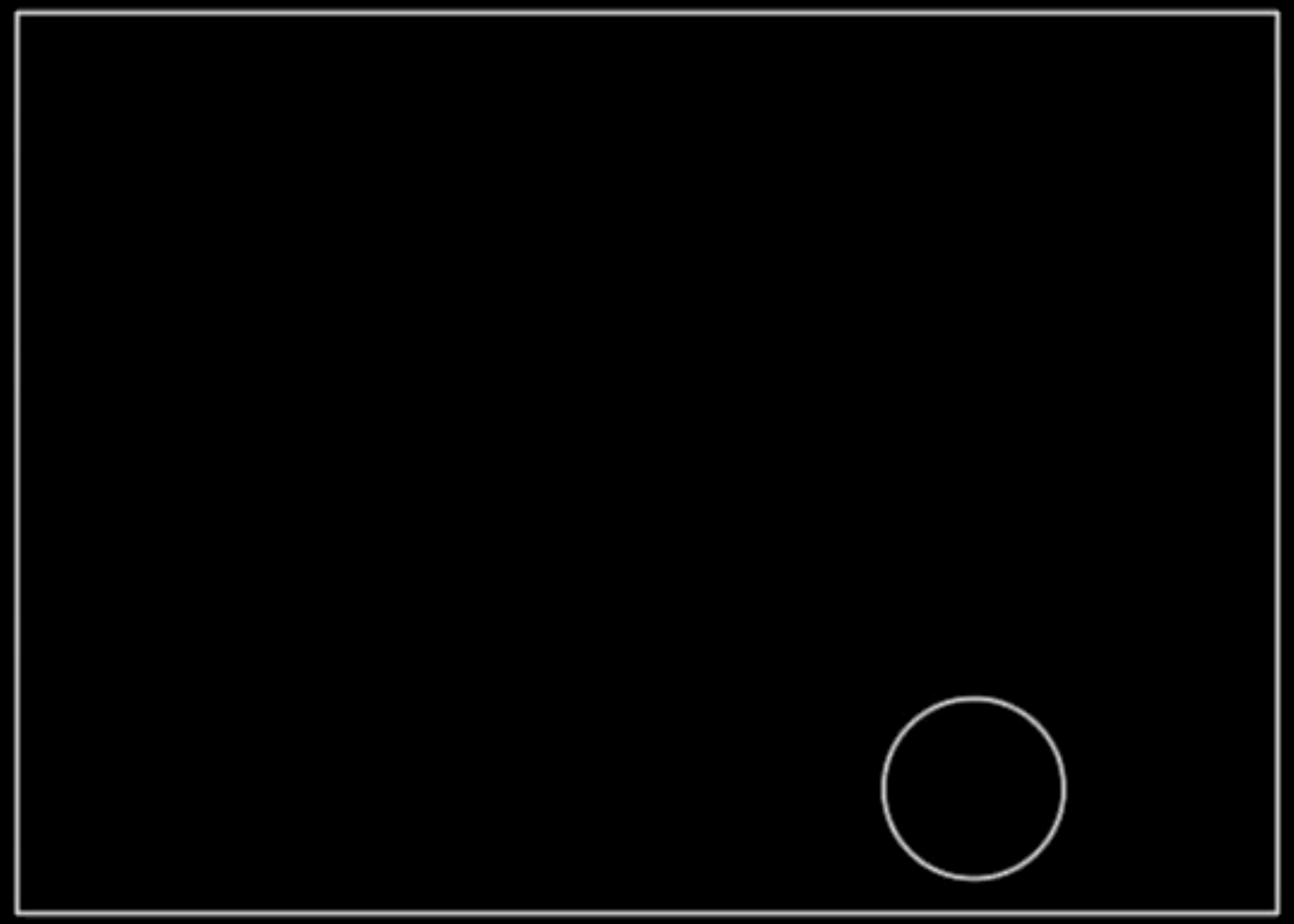
```
SKRange* range = [SKRange rangeWithLowerLimit:-100.0f upperLimit:100.0f]];

SKConstraint* constraintX = [SKConstraint positionX:range]; //X constraint
SKConstraint* constraintY = [SKConstraint positionY:range]; //Y constraint

node.constraints = @[constraintX,constraintY];
```



Example

Position constraint

```
SKRange* range = [SKRange rangeWithLowerLimit:-100.0f upperLimit:100.0f]];

SKConstraint* constraintX = [SKConstraint positionX:range]; //X constraint
SKConstraint* constraintY = [SKConstraint positionY:range]; //Y constraint

node.constraints = @[constraintX,constraintY];
```



Constraints

Summary

Removes boilerplate code in scene update

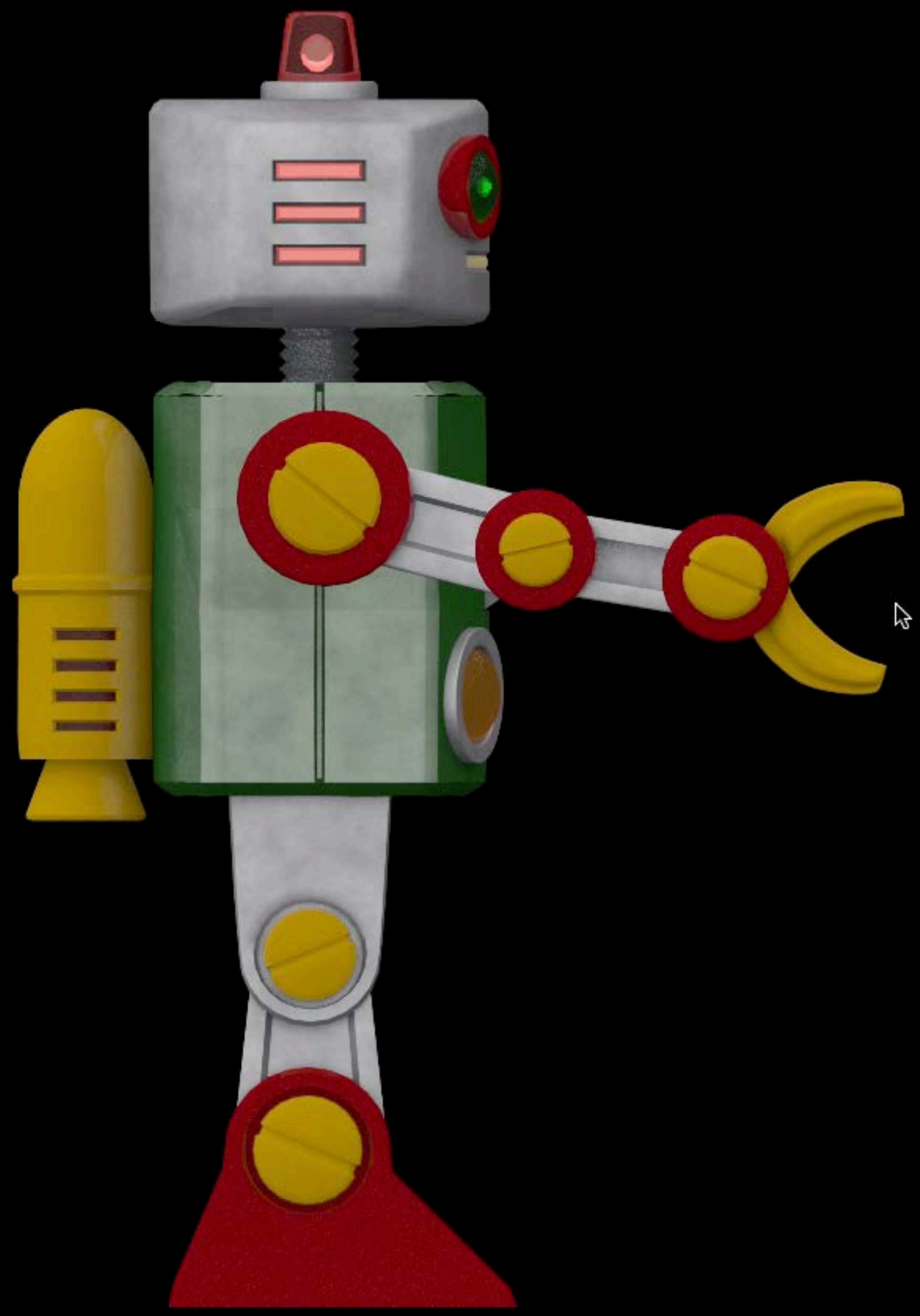
Evaluated in order list in the array

Offers a variety of constraints

Inverse Kinematics

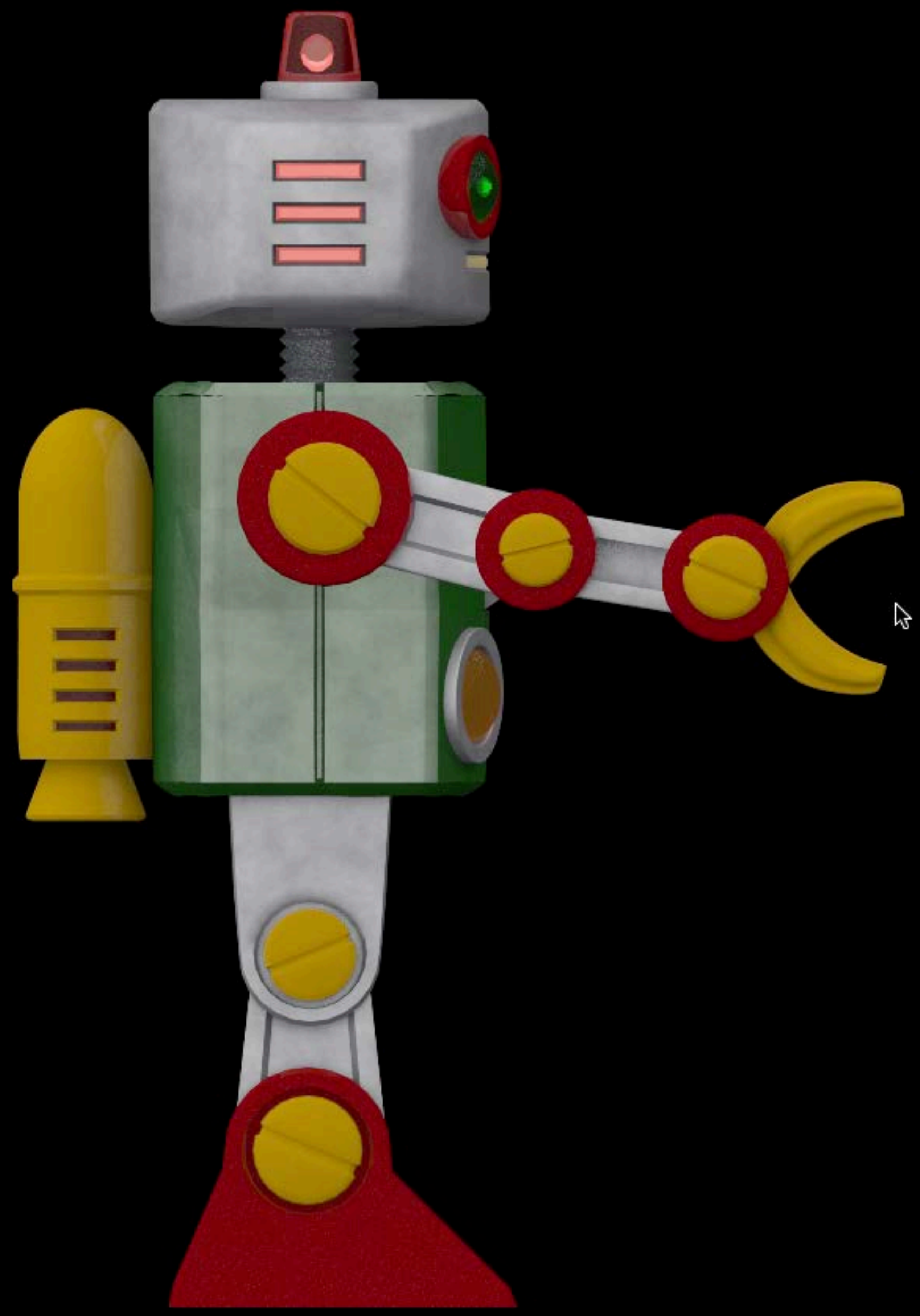
Inverse Kinematics

Introduction



Inverse Kinematics

Introduction

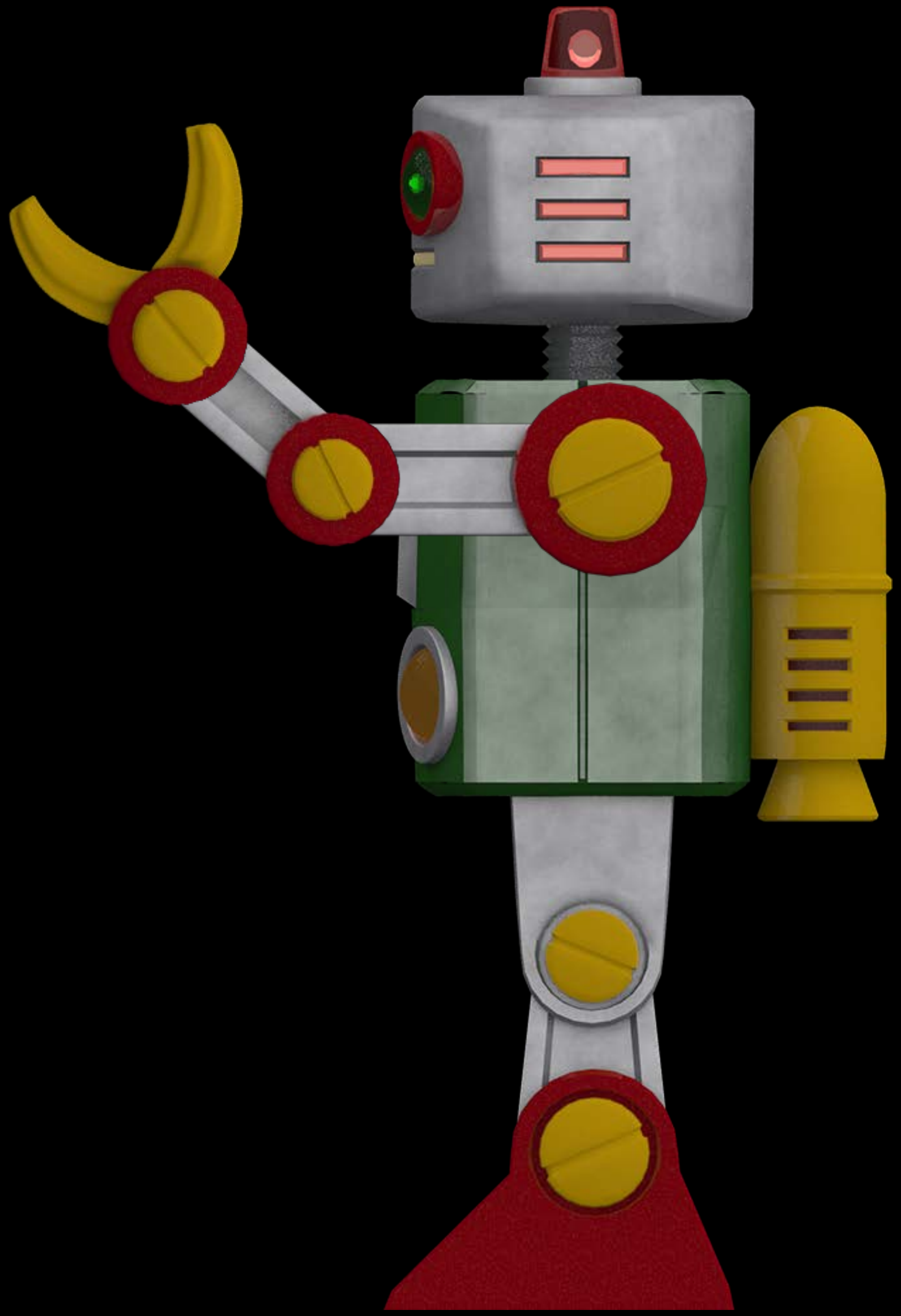


Inverse Kinematics

Joint hierarchy definition

Use sprites to represent joints

- Anchor points
- DOF constraints
- Reach to point

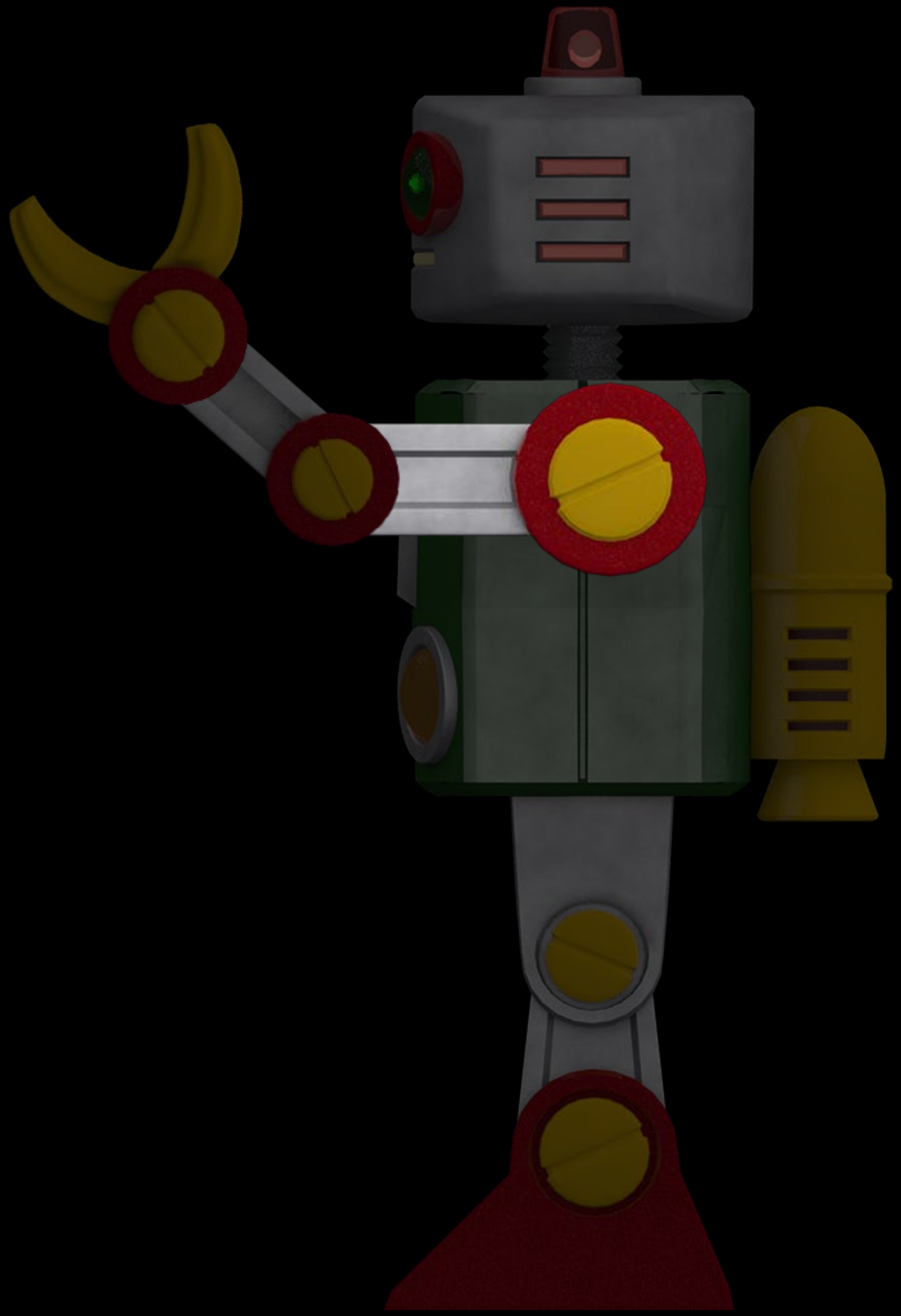


Inverse Kinematics

Joint hierarchy definition

Use sprites to represent joints

- Anchor points
- DOF constraints
- Reach to point

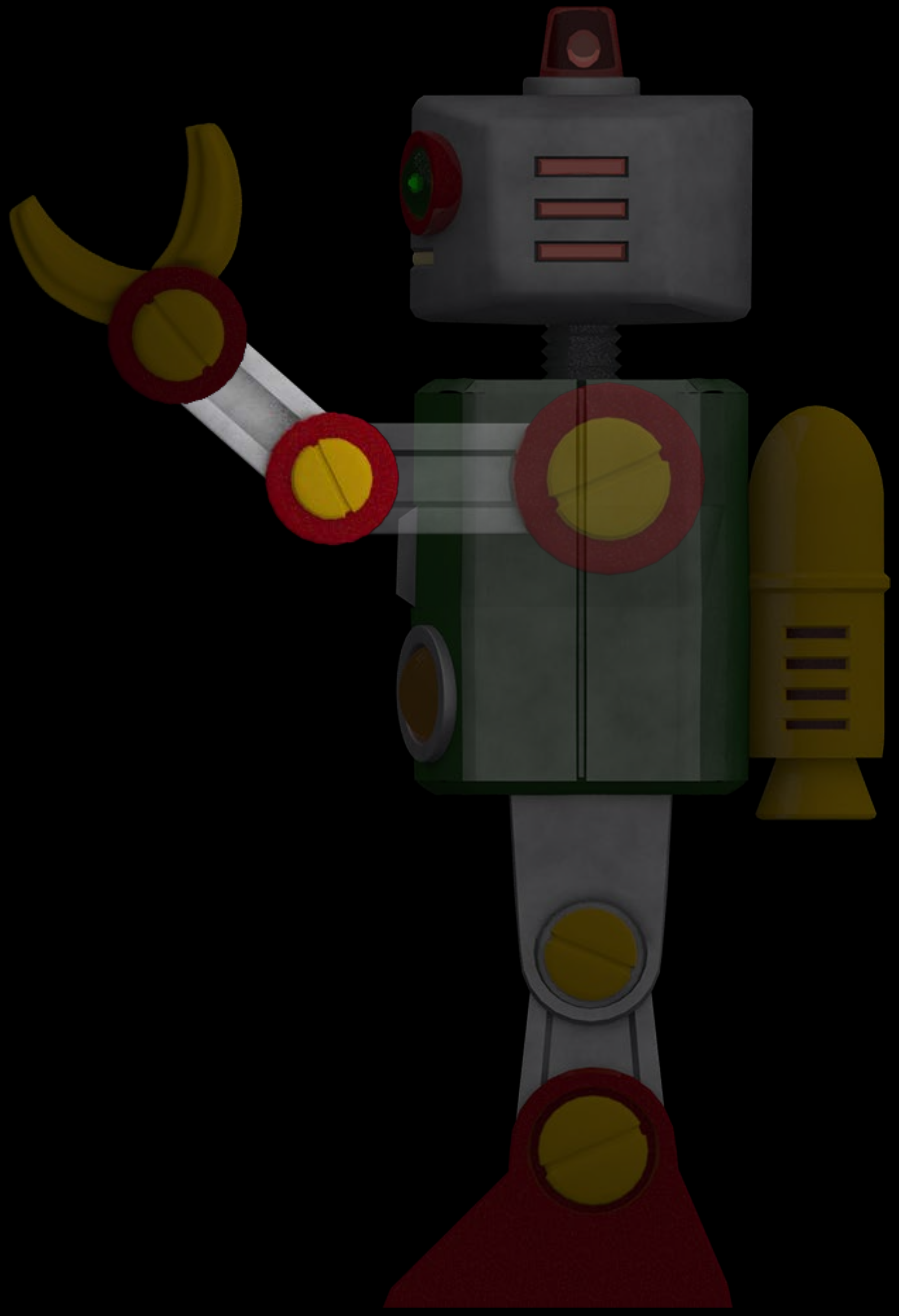


Inverse Kinematics

Joint hierarchy definition

Use sprites to represent joints

- Anchor points
- DOF constraints
- Reach to point

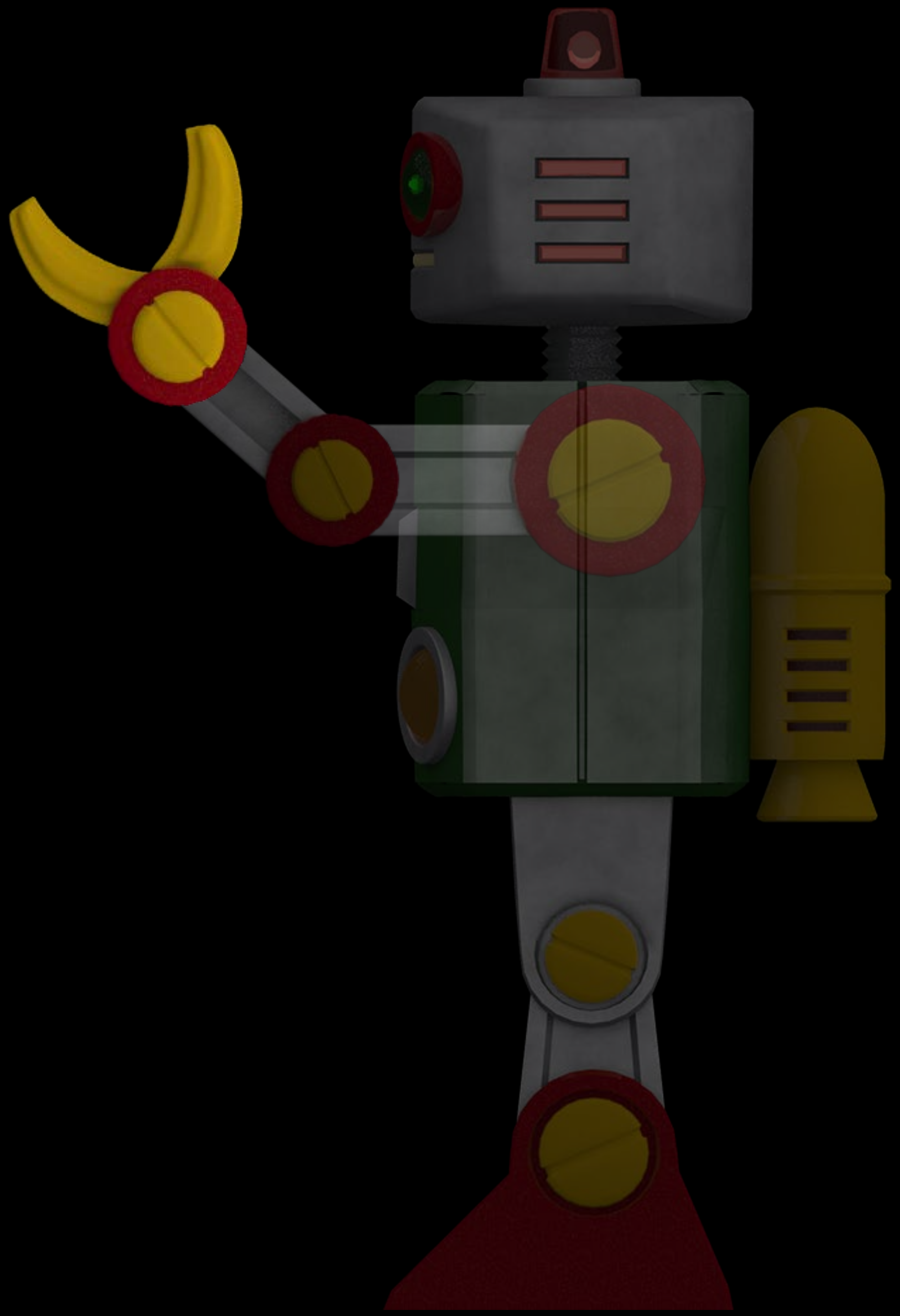


Inverse Kinematics

Joint hierarchy definition

Use sprites to represent joints

- Anchor points
- DOF constraints
- Reach to point



Inverse Kinematics in SpriteKit

Joints defined by scene graph

Existing parent-child relationship

Defines IK constraints on each joint

Controls the minimum and maximum rotation

Joint rotates around its anchor point

SKReachConstraints

```
@property (nonatomic, assign) CGFloat lowerAngleLimit;
@property (nonatomic, assign) CGFloat upperAngleLimit;
```

To set the constraint on a node, use

```
@property (nonatomic, copy) SKReachConstraints *reachConstraints;
```

Inverse Kinematics

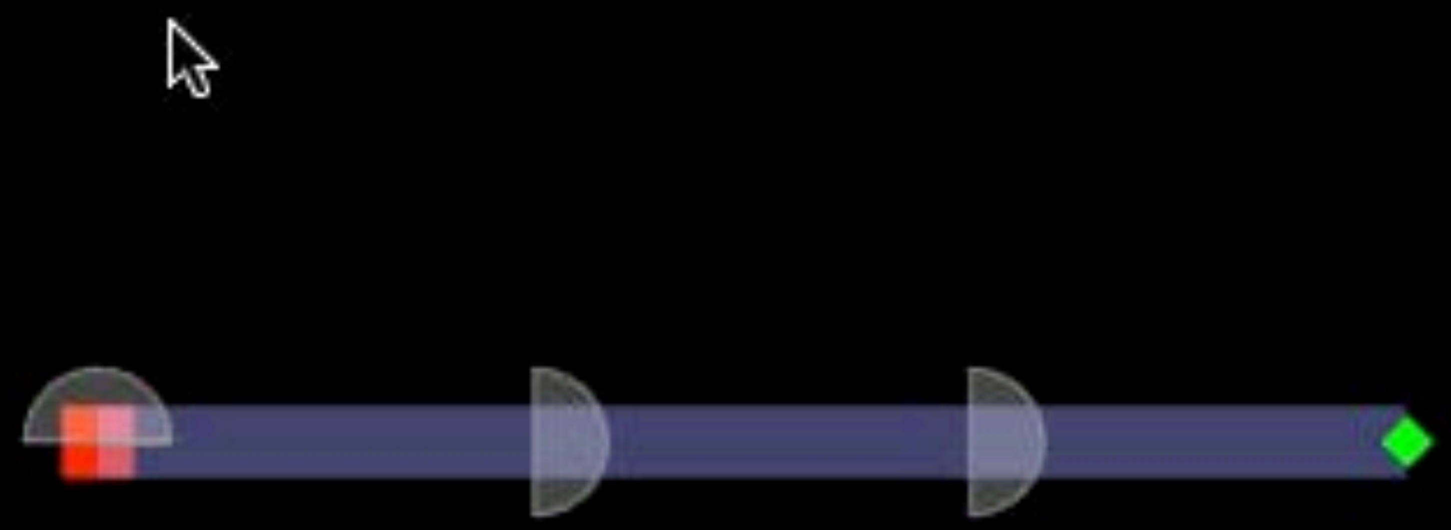
Driving the joints

```
+ (SKAction *)reachTo:(CGPoint)position
             rootNode:(SKNode *)root
             duration:(NSTimeInterval)sec;

+ (SKAction *)reachToNode:(SKNode *)node
                 rootNode:(SKNode *)root
                 duration:(NSTimeInterval)sec;
```

Inverse Kinematics

Example

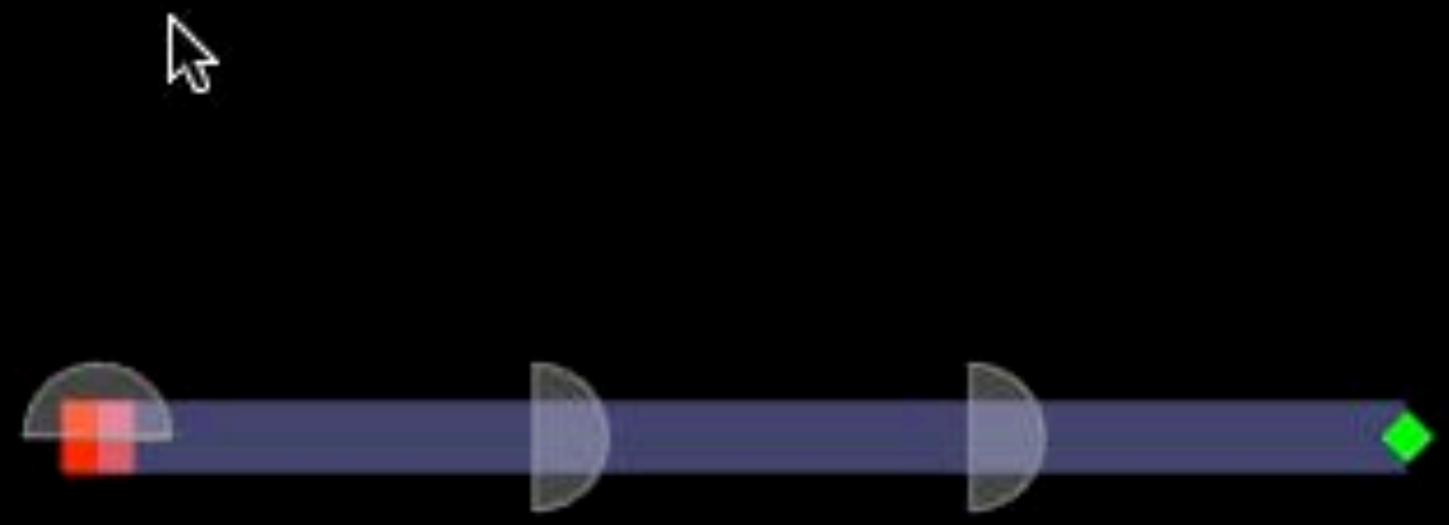
```
[_endEffector runAction:[SKAction reachTo:p
                                 rootNode:_root
                                 duration:0.2]];
```



Inverse Kinematics

Example

```
[_endEffector runAction:[SKAction reachTo:p
                                 rootNode:_root
                                 duration:0.2]];
```



Inverse Kinematics

3D

Shared IK solver is also available in SceneKit

`SCNIKConstraint`

`SCNNode.constraints`

`SCNNode.influenceFactor`

Inverse Kinematics

3D

Shared IK solver is also available in SceneKit

`SCNIKConstraint`

`SCNNode.constraints`

`SCNNode.influenceFactor`

Inverse Kinematics

3D

Shared IK solver is also available in SceneKit

`SCNIKConstraint`

`SCNNode.constraints`

`SCNNode.influenceFactor`

Inverse Kinematics

3D

Shared IK solver is also available in SceneKit

`SCNIKConstraint`

`SCNNode.constraints`

`SCNNode.influenceFactor`

Inverse Kinematics

Summary

Uses existing scene graph to represent hierarchy

Constraints can be set on each joint

Easy to use action to reach for position or node

Physics Fields

Physics Fields

Introduction

Physics Fields

Introduction

Physics Fields

Overview

Fields simulate physical forces

Fields affect physics bodies in a region

Variety of field types

- Drag field, vortex field, radial gravity field, linear gravity field, velocity field, noise field, turbulence field, spring field, electric field, magnetic field

Physics Fields

Field updates

The field node is in the scene's node tree

Physics bodies exist in the scene's node tree

Physics bodies are located inside the field node's region

Field's `categoryBitMask` property and the physics body's `fieldBitMask`

Physics Fields

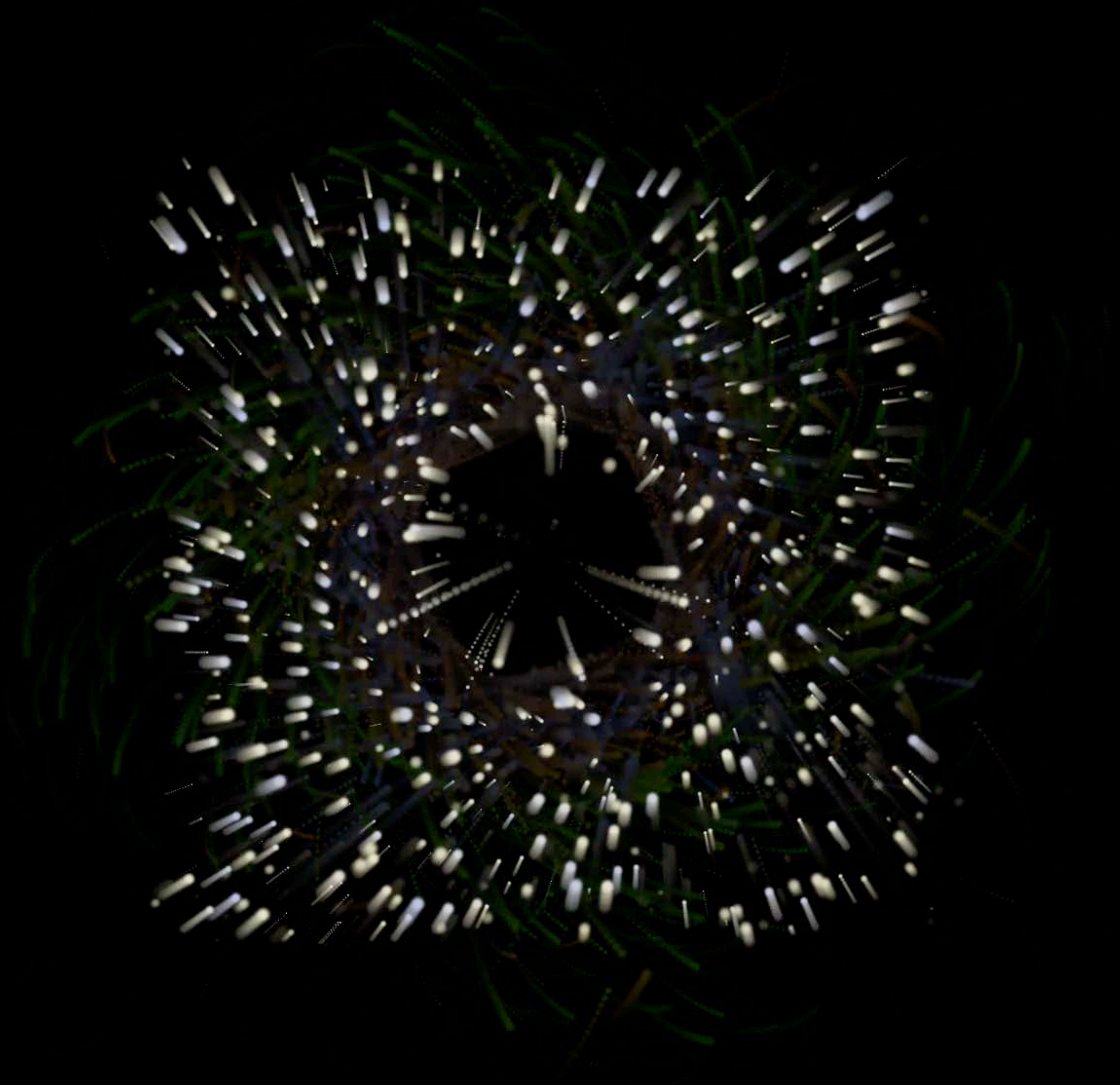
Field node

SKFieldNode

- `region`
- `strength`
- `minRadius`
- `falloff`
- `categoryBitMask`

SKPhysicsBody

- `fieldBitMask`



Physics Fields

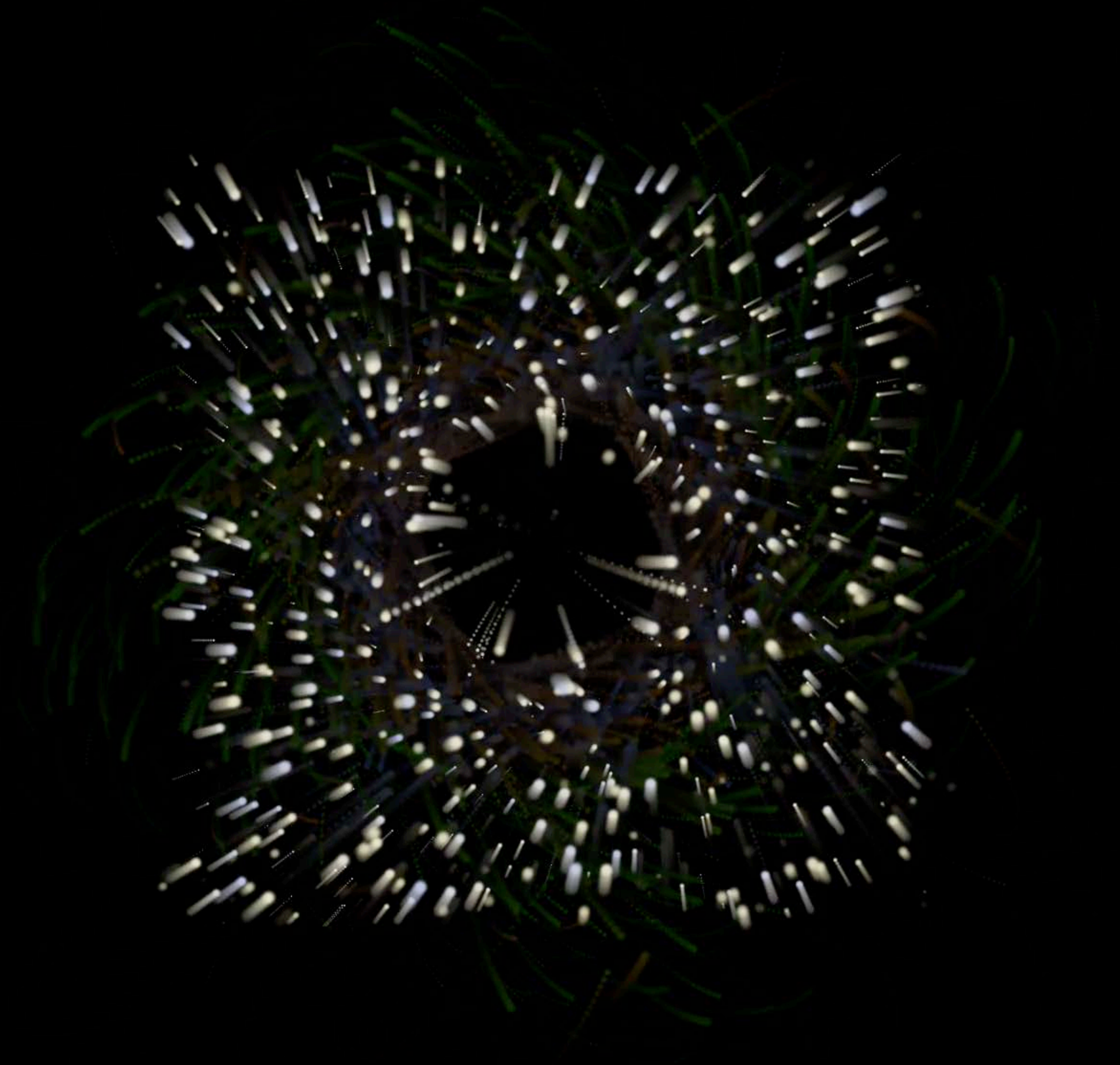
Field node

SKFieldNode

- region
- strength
- minRadius
- falloff
- categoryBitMask

SKPhysicsBody

- fieldBitMask



Physics Fields

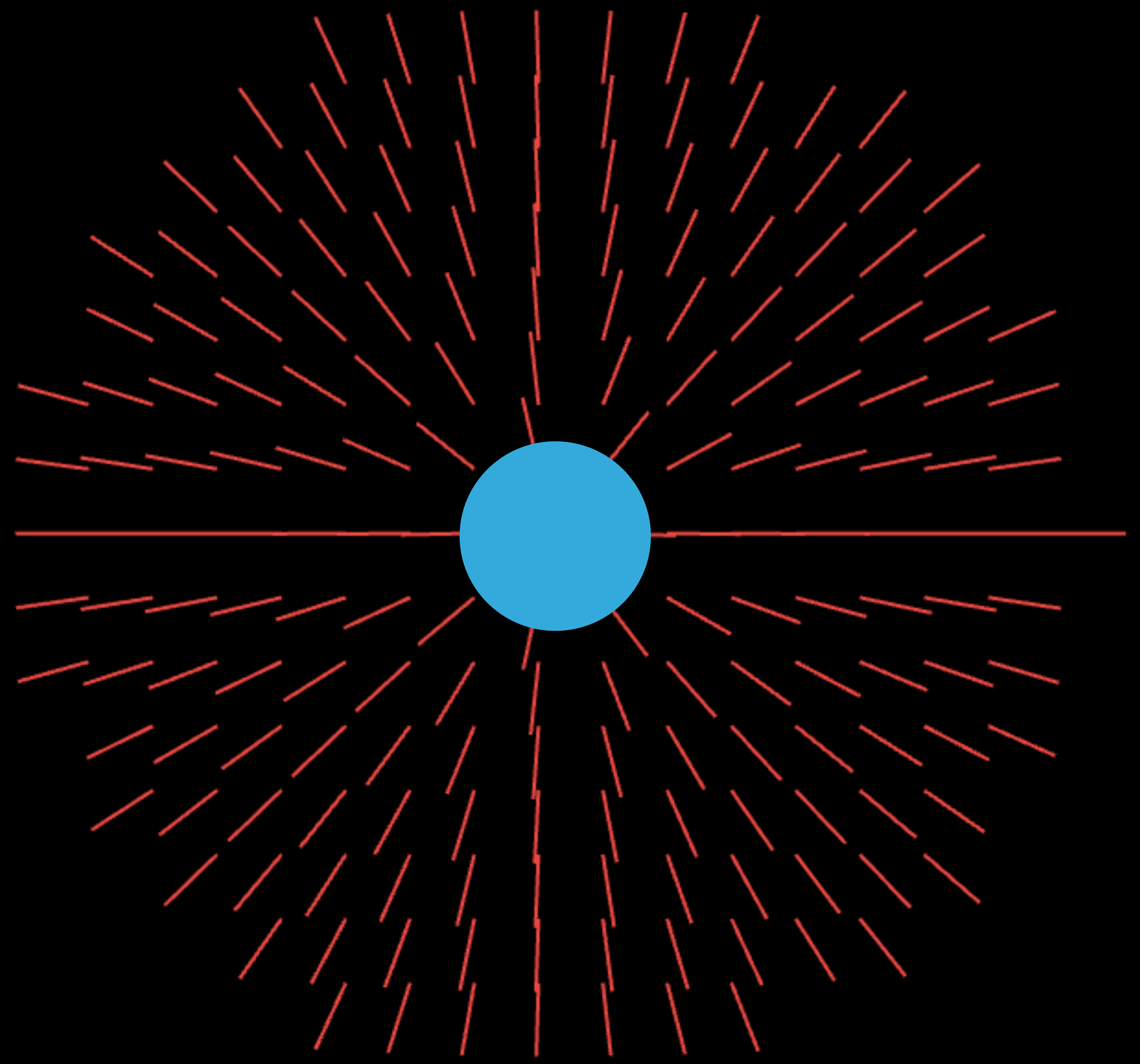
Region

New object—SKRegion

Regions define a 2D space

Can be infinite, rectangle, circle, CGPath

Can invert, subtract, union, intersect regions



Physics Fields

Particle interactions

Particle interactions

- fieldBitMask property

Physics Fields

Particle interactions

Particle interactions

- fieldBitMask property

Physics Fields

Linear gravity field

↓

Applies force in a given direction

Physics Fields

Linear gravity field

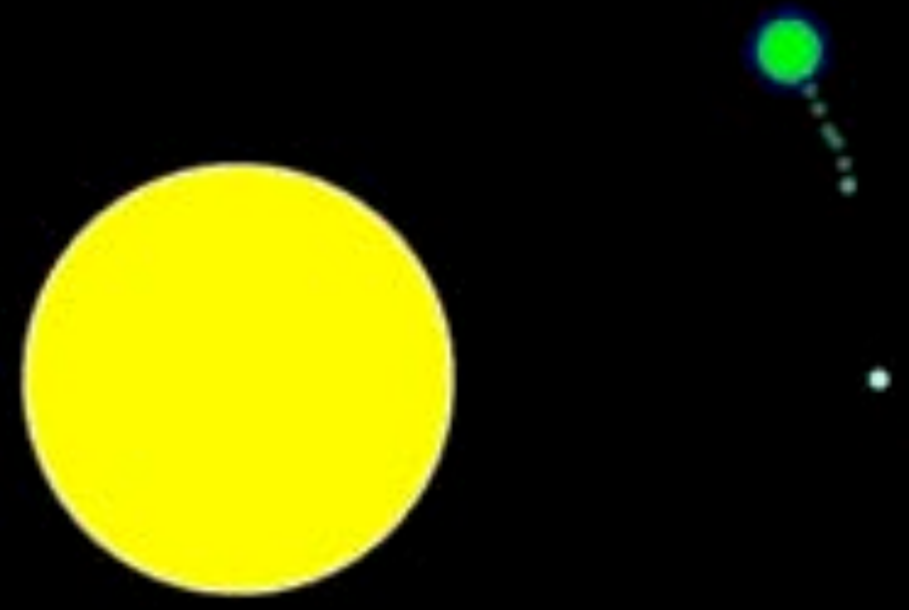
↓

Applies force in a given direction

Physics Fields

Radial gravity field

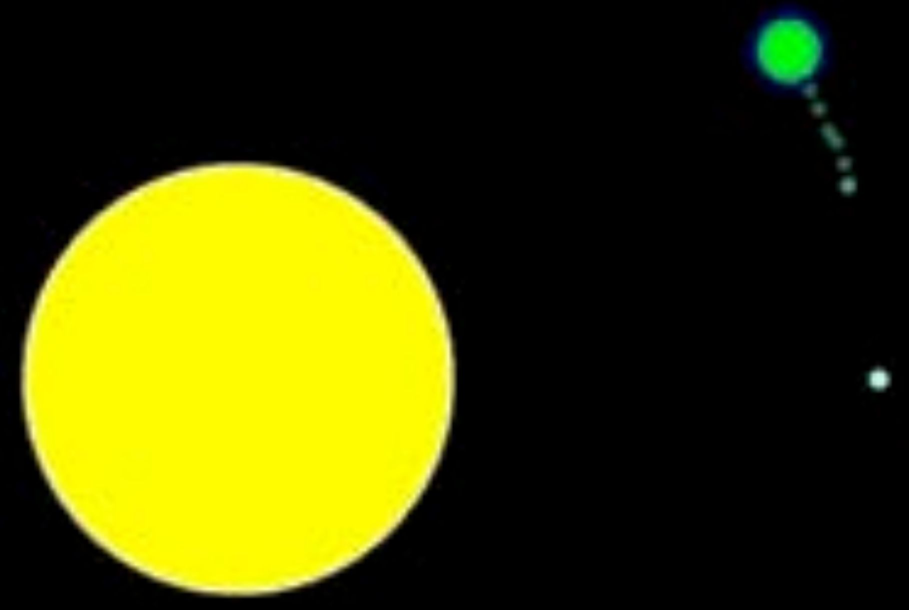
Applies force toward field origin



Physics Fields

Radial gravity field

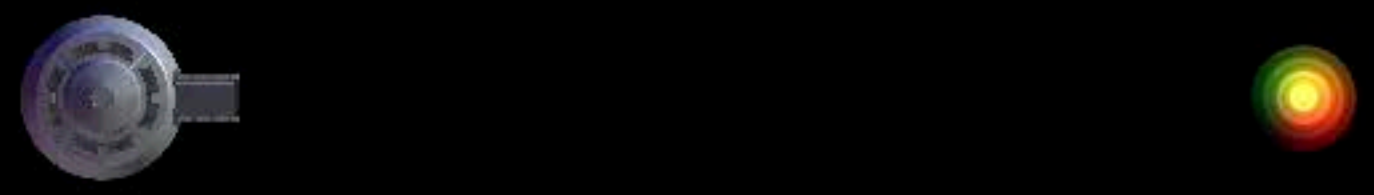
Applies force toward field origin



Physics Fields

Spring field

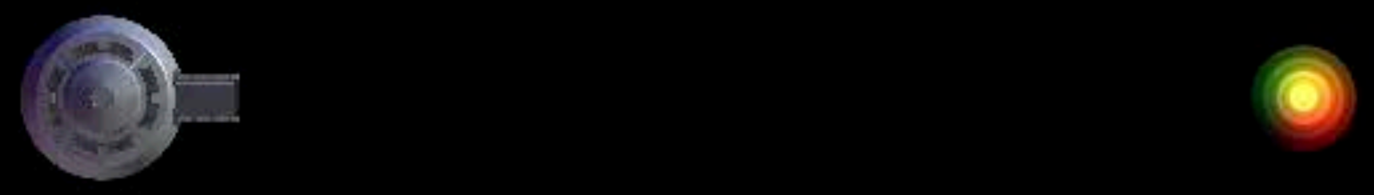
Applies force toward field origin with spring oscillation



Physics Fields

Spring field

Applies force toward field origin with spring oscillation



Physics Fields

Noise field

Applies a noisy, random force

Physics Fields

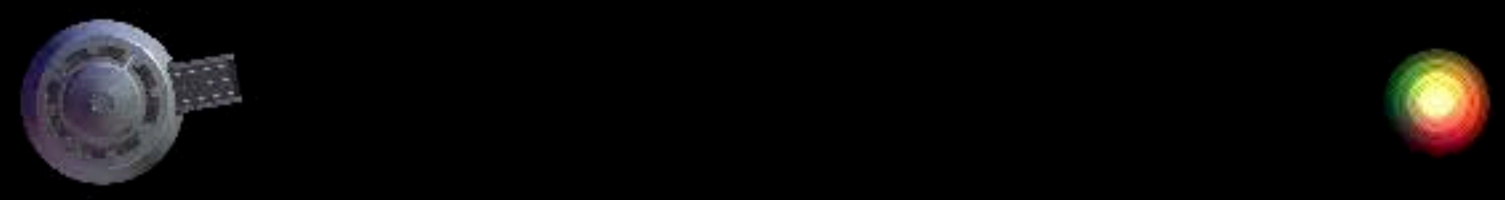
Noise field

Applies a noisy, random force

Physics Fields

Electric field

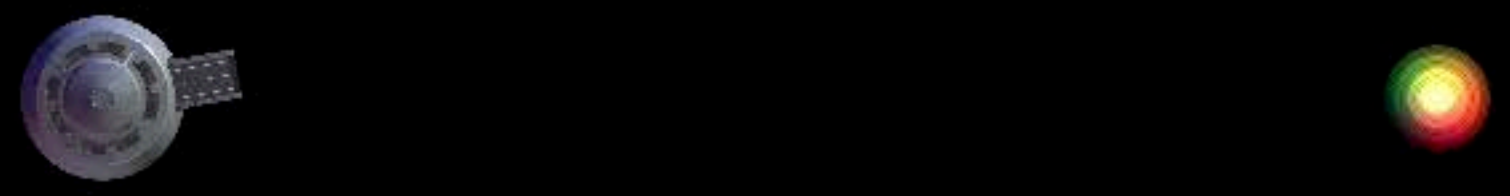
Applies a force proportional an object’s charge



Physics Fields

Electric field

Applies a force proportional an object's charge



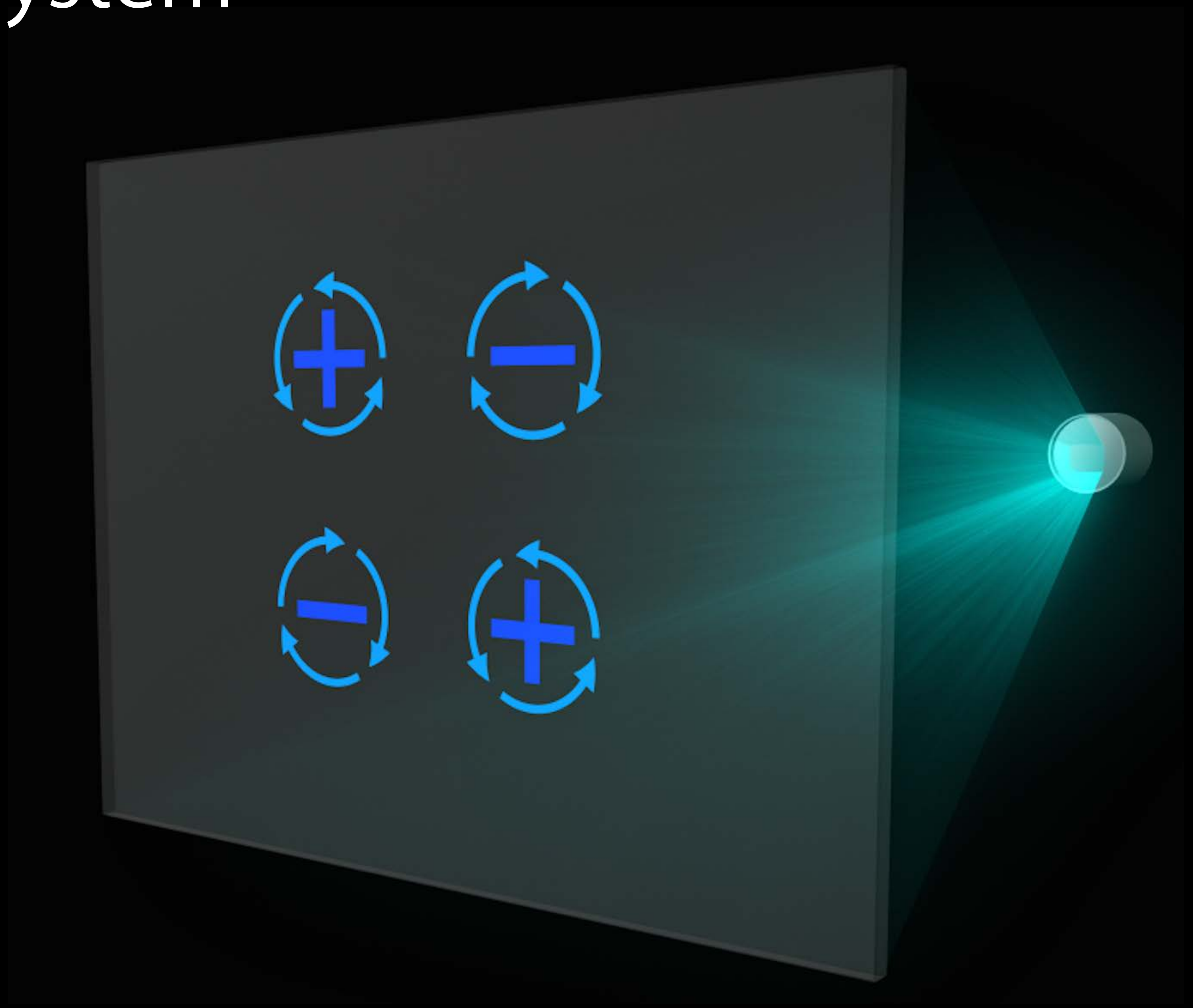
Physics Fields

Interactions

Physics fields are provided as building blocks

Objects can interact with multiple fields

Lorenz system



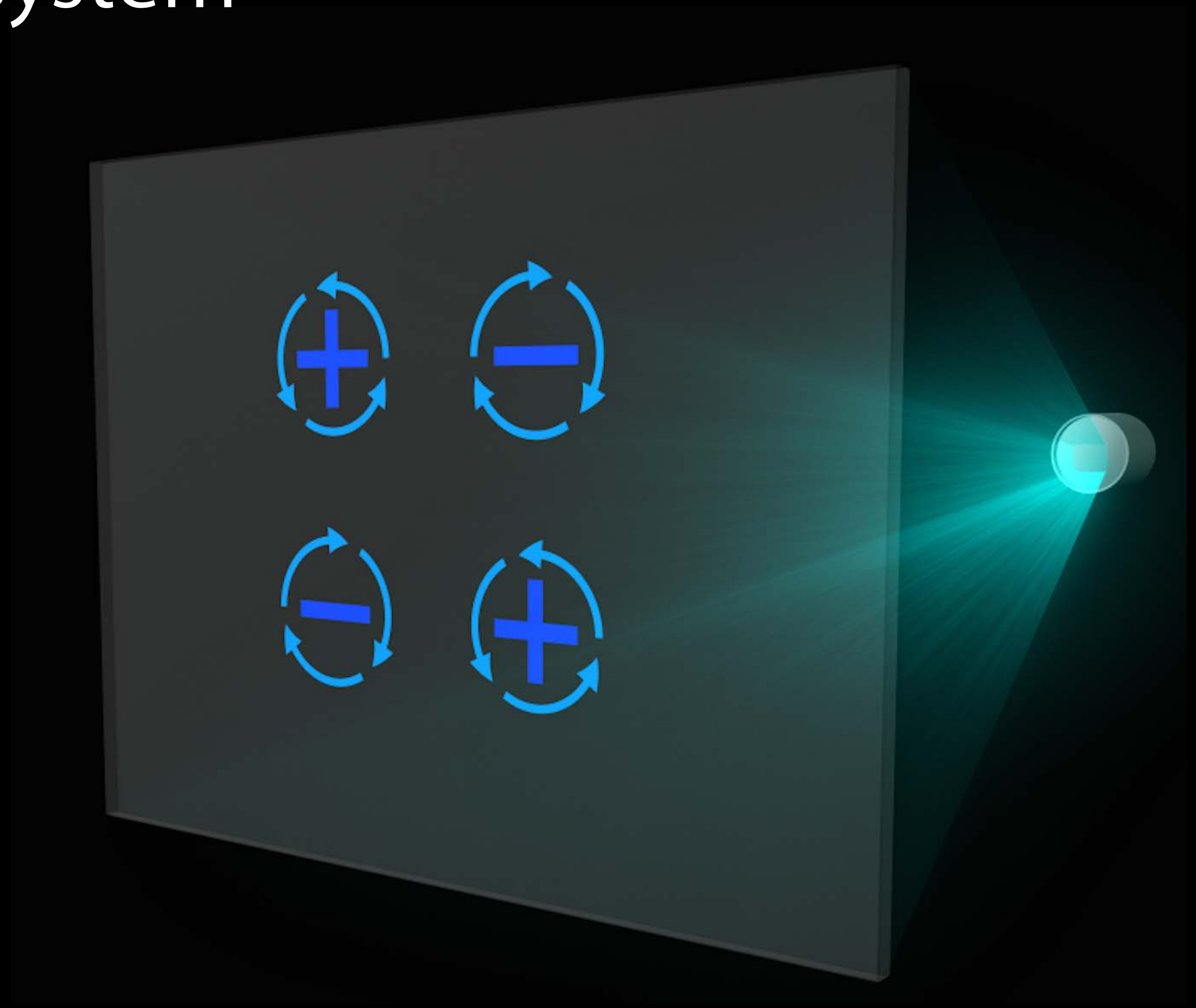
Physics Fields

Interactions

Physics fields are provided as building blocks

Objects can interact with multiple fields

Lorenz system



Physics Fields

Summary

Fields are fast and efficient

Interact with physics bodies and particles

Interact with other physics fields

Shaders

Lighting and Shadows

New Physics

Integration with SceneKit

Tools

Improvements

Shaders

Lighting and Shadows

New Physics

Integration with SceneKit

Tools

Improvements

SpriteKit and SceneKit

Introduction

SpriteKit and SceneKit

Introduction

SceneKit Integration

Include 3D content in SpriteKit games

Control 3D objects like regular SKNodes

Renders as part of SpriteKit scene graph

Loosely coupled, deeply integrated

SK3DNode

Overview

Incorporate 3D content into a SpriteKit-based game

Attaches an SCNScene to a SKNode

- Renders 3D content directly inside SpriteKit GL context

Add existing .dae or .abc file to SKScene

Use the `scnScene` to specify the 3D scene to be rendered

SK3DNode

Creation

scnNodeWithViewportSize

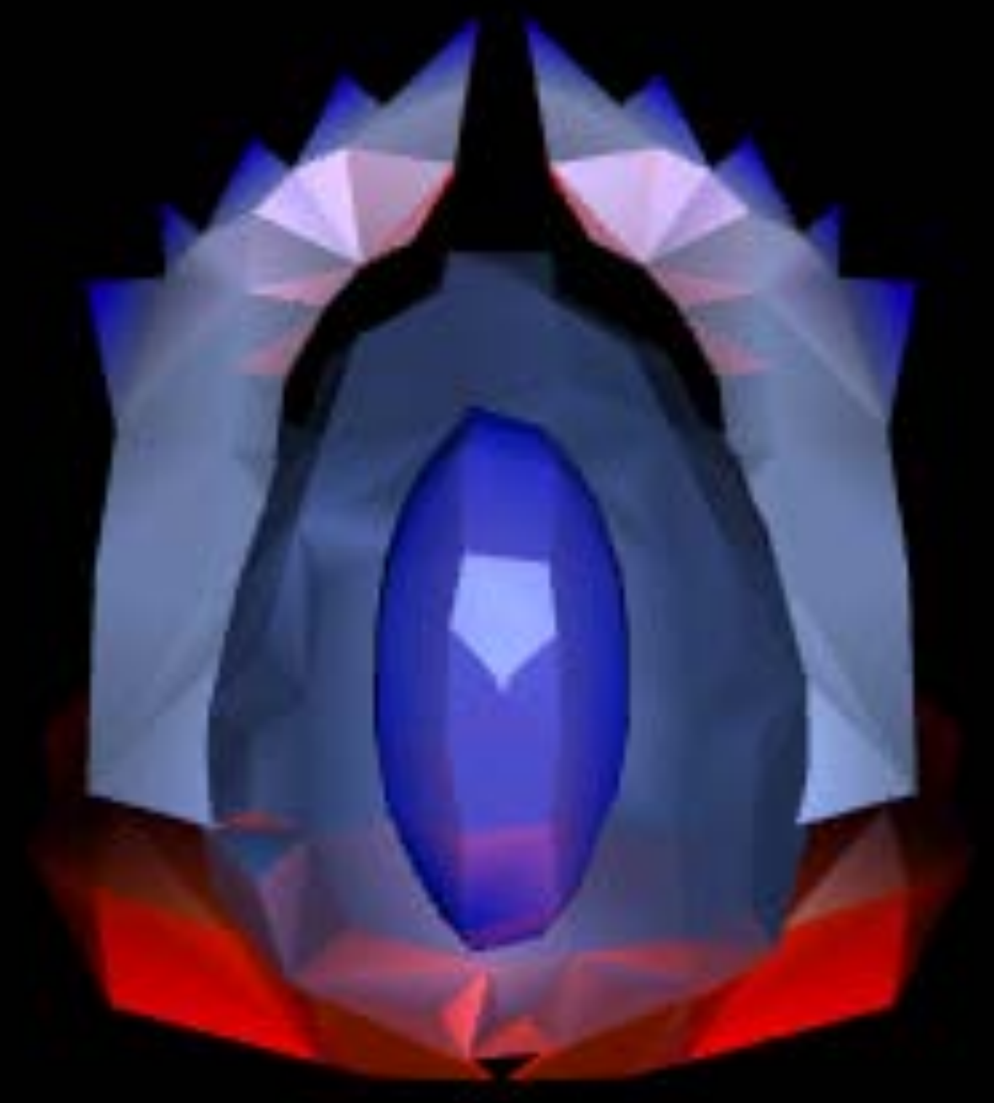
- Creates and initializes a new 3D node

```
@property SCNScene *scnScene
@property CGSize viewportSize
@property(nonatomic, retain) SCNNode *pointOfView
@property(nonatomic) BOOL autoenablesDefaultLighting
```

SK3DNode

Example

```
SK3DNode *alien3D = [[SK3DNode alloc] initWithViewportSize:CGSizeMake(200,
200)];
SCNScene *alienSCN = [SCNScene sceneNamed:@"alien.dae"];
alien3D.scnScene = alienSCN;
[self addChild:alien3D];
```



SceneKit Integration

Textures and sounds

Use SKTexture with SceneKit objects

- SKTextureAtlas and generation tool
- Automatic normal map generation

Shared auto playback interface

SceneKit Integration

Summary

SceneKit Integration

Summary

Shaders

Lighting and Shadows

New Physics

Integration with SceneKit

Tools

Improvements

Shaders

Lighting and Shadows

New Physics

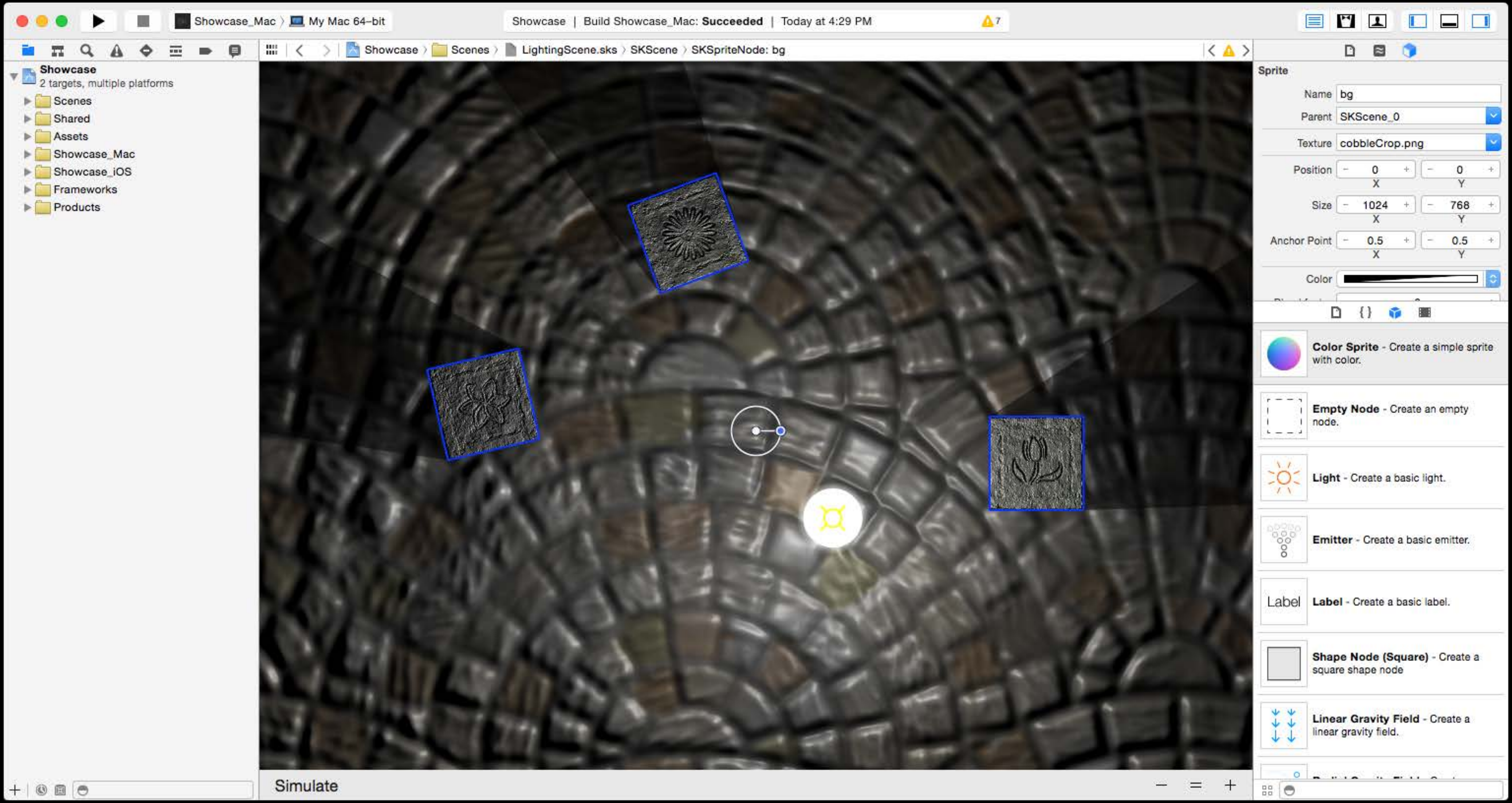
Integration with SceneKit

Tools

Improvements

SpriteKit Editor

Introduction



Showcase | Build Showcase_Mac: **Succeeded** | Today at 4:29 PM

Showcase › Scenes › LightingScene.sks › SKScene › SKSpriteNode: bg

Sprite

Name: bg

Parent: SKScene_0

Texture: cobbleCrop.png

Position: X 0, Y 0

Size: X 1024, Y 768

Anchor Point: X 0.5, Y 0.5

Color

Color Sprite - Create a simple sprite with color.

Empty Node - Create an empty node.

Light - Create a basic light.

Emitter - Create a basic emitter.

Label **Label** - Create a basic label.

Shape Node (Square) - Create a square shape node

Linear Gravity Field - Create a linear gravity field.

SpriteKit Editor

Overview

Provides various editors integrated inside of Xcode

Data drive your game

Updated and simplified game project templates

Exported content can be deployed on OS X and iOS

Debug and edit existing scenes

```
[NSKeyedArchiver archiveRootObject:self toFile:@"snapshot.sks"];
```

SpriteKit Editor

Features

Object manipulation and placement

Physics bodies set-up

3D node

Shading and lighting

Inverse kinematic

Shader editor

Demo

Shaders

Lighting and Shadows

New Physics

Integration with SceneKit

Tools

Improvements

Shaders

Lighting and Shadows

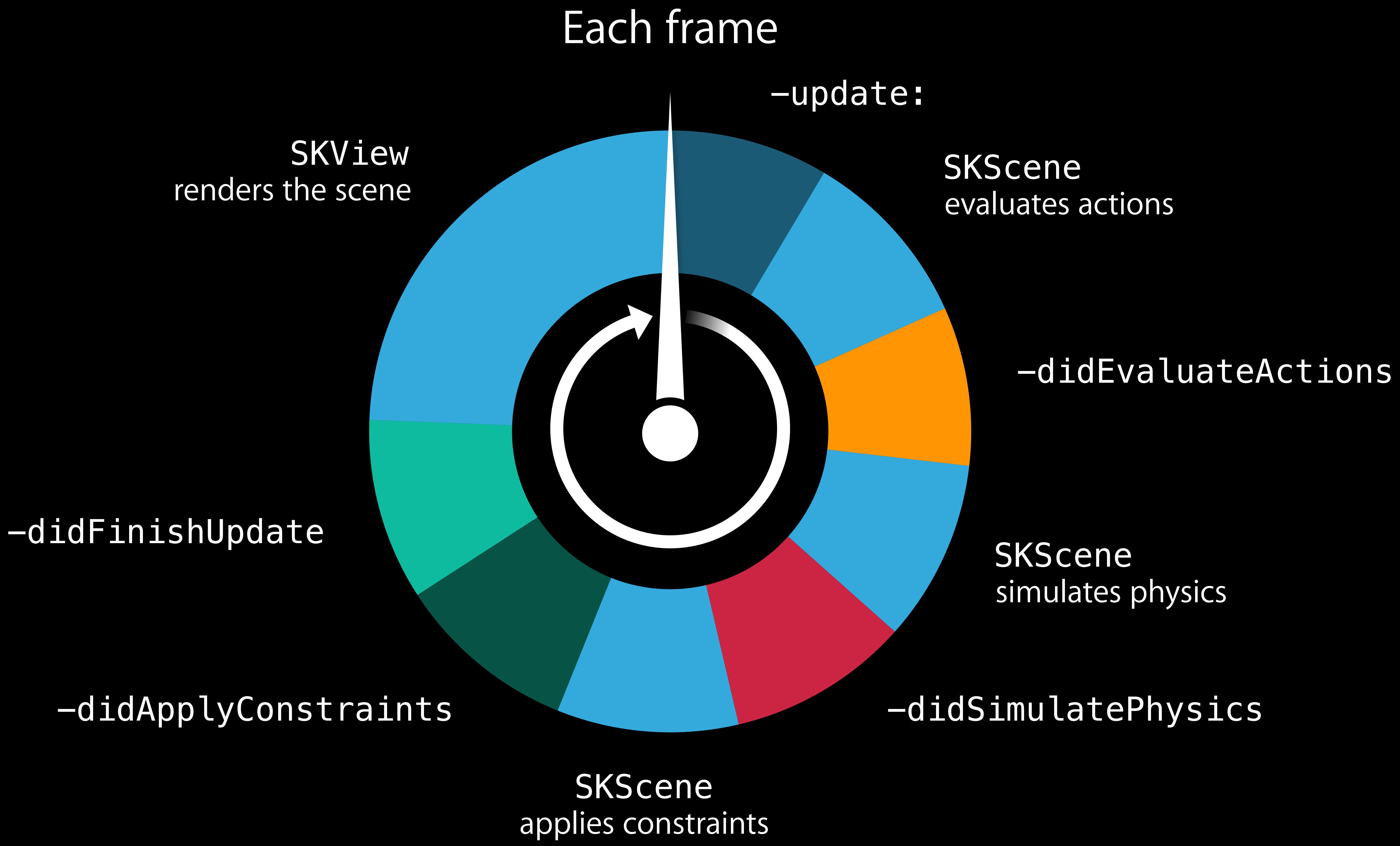
New Physics

Integration with SceneKit

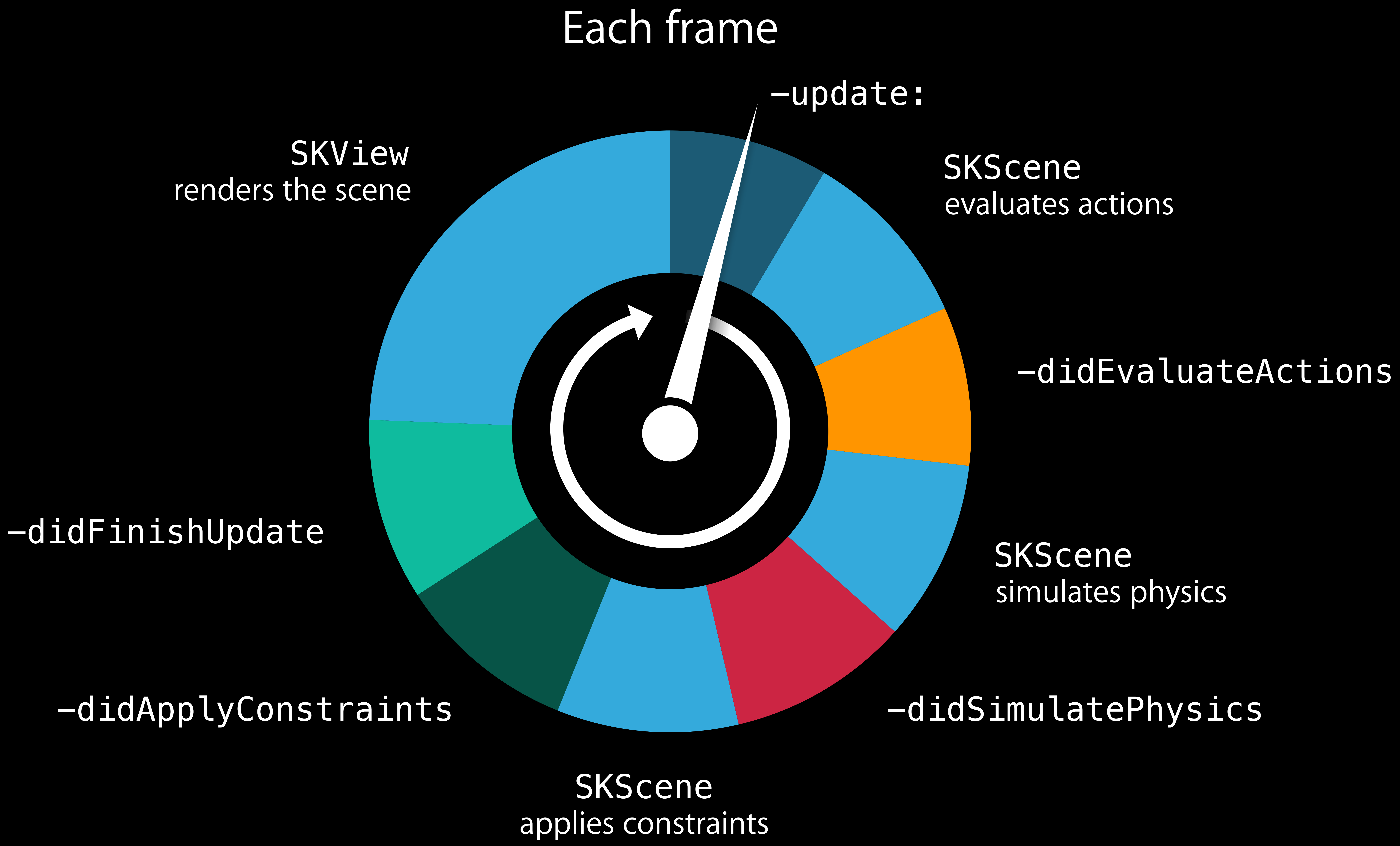
Tools

Improvements

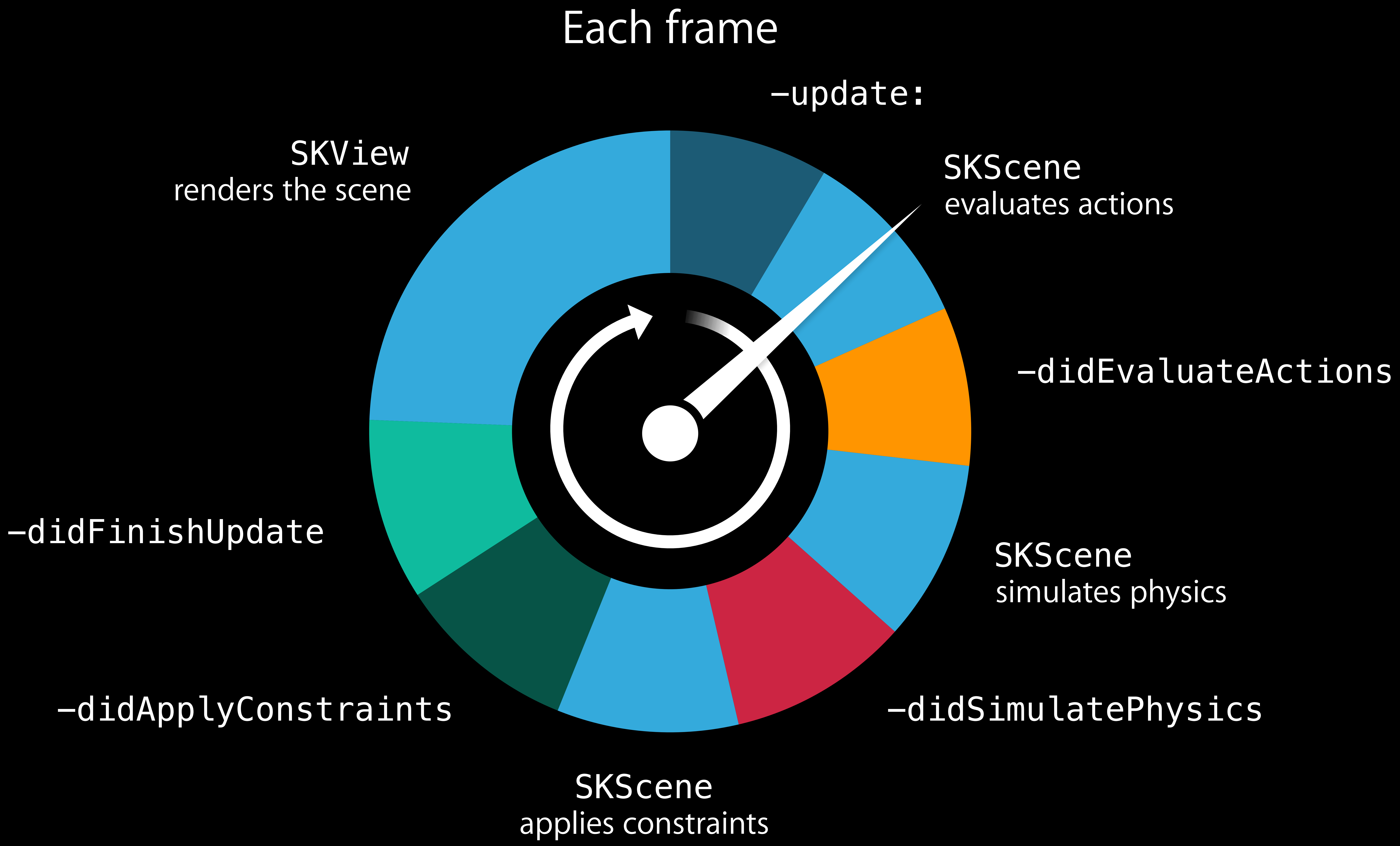
SKScene


Each frame
-update:
SKScene
evaluates actions
-didEvaluateActions
SKScene
simulates physics
-didSimulatePhysics
SKScene
applies constraints
-didApplyConstraints
-didFinishUpdate
SKView
renders the scene

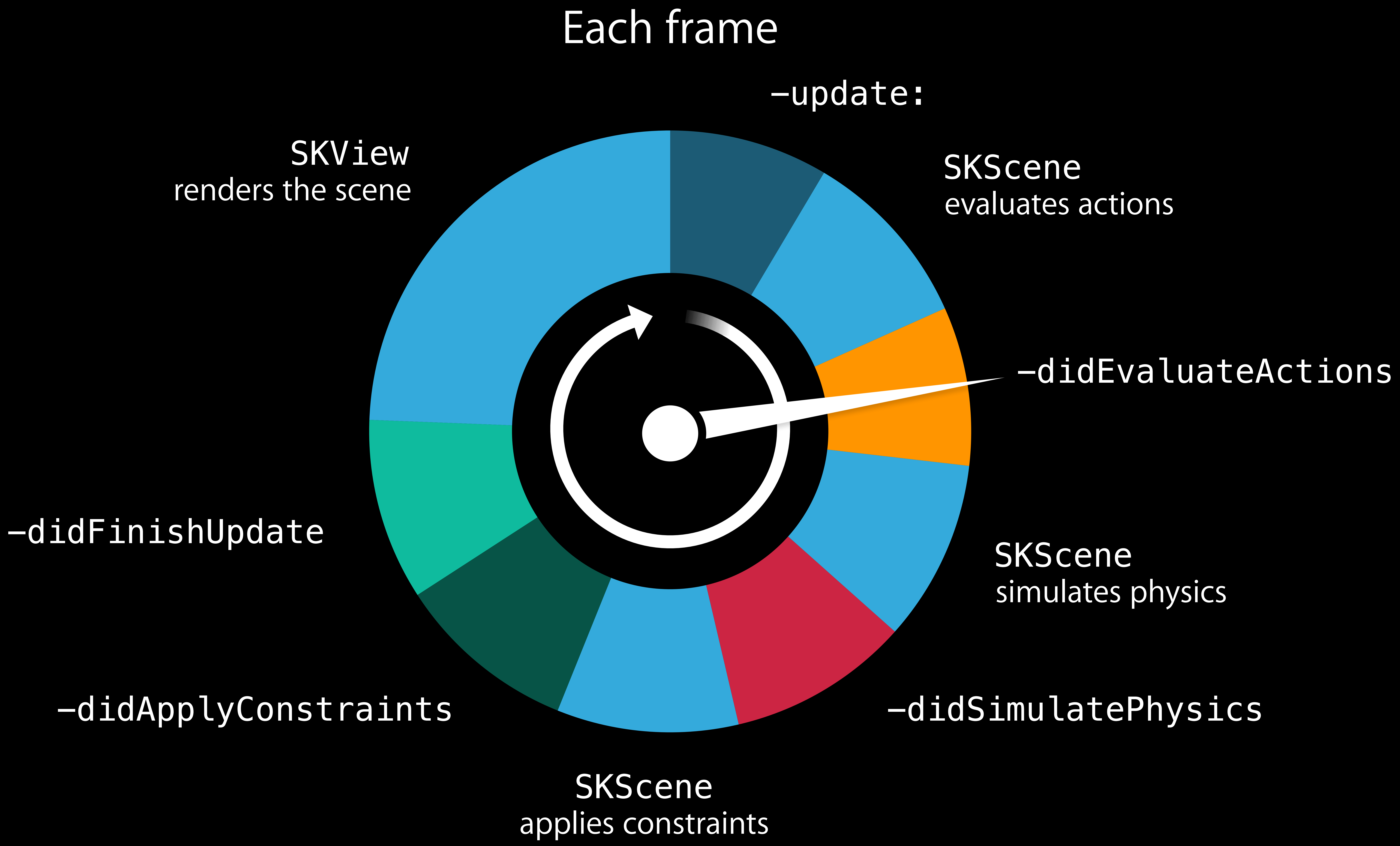
SKScene


Each frame
-update:
SKScene
evaluates actions
-didEvaluateActions
SKScene
simulates physics
-didSimulatePhysics
SKScene
applies constraints
-didApplyConstraints
-didFinishUpdate
SKView
renders the scene

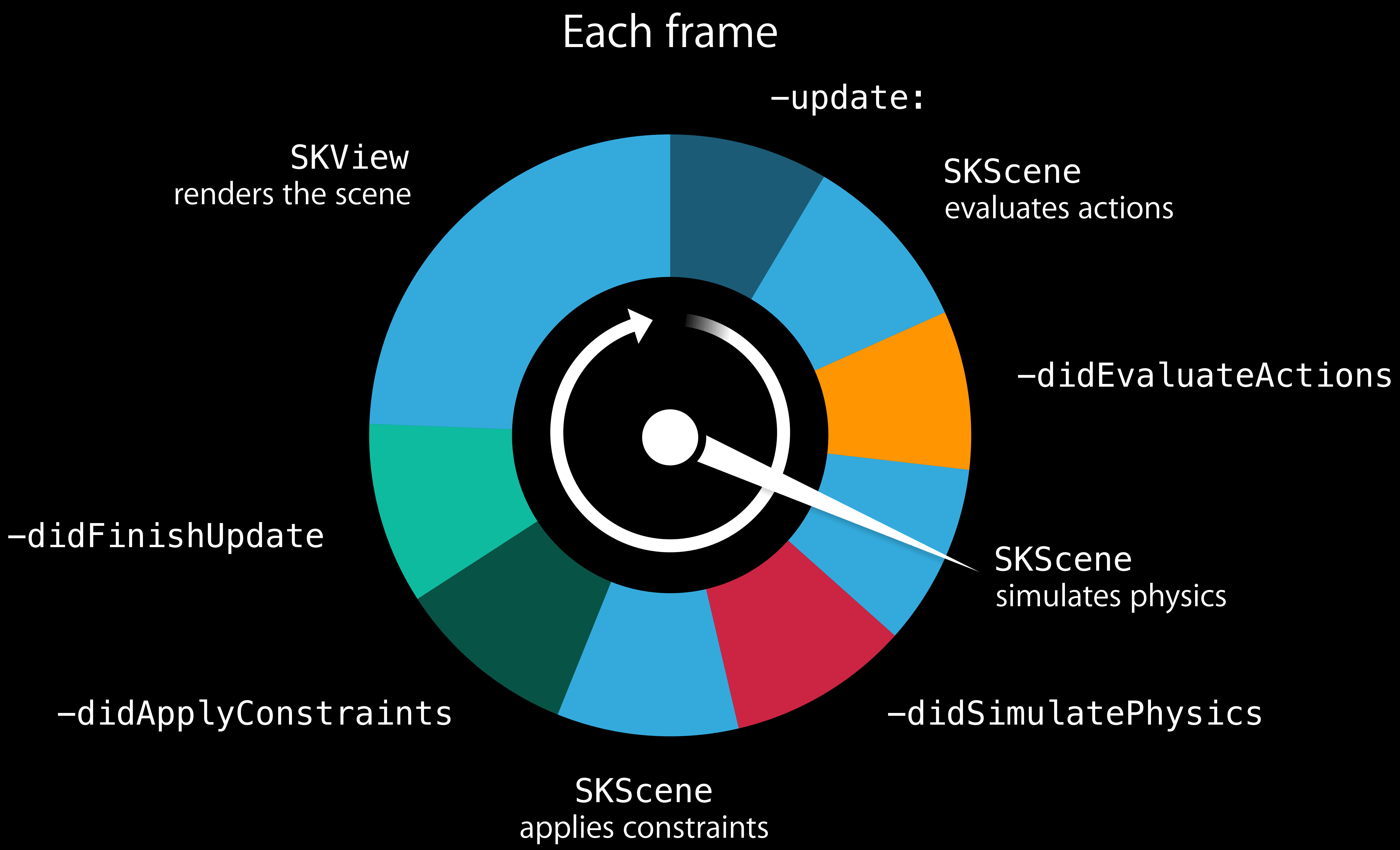
SKScene


Each frame
-update:
SKScene
evaluates actions
-didEvaluateActions
SKScene
simulates physics
-didSimulatePhysics
SKScene
applies constraints
-didApplyConstraints
-didFinishUpdate
SKView
renders the scene

SKScene

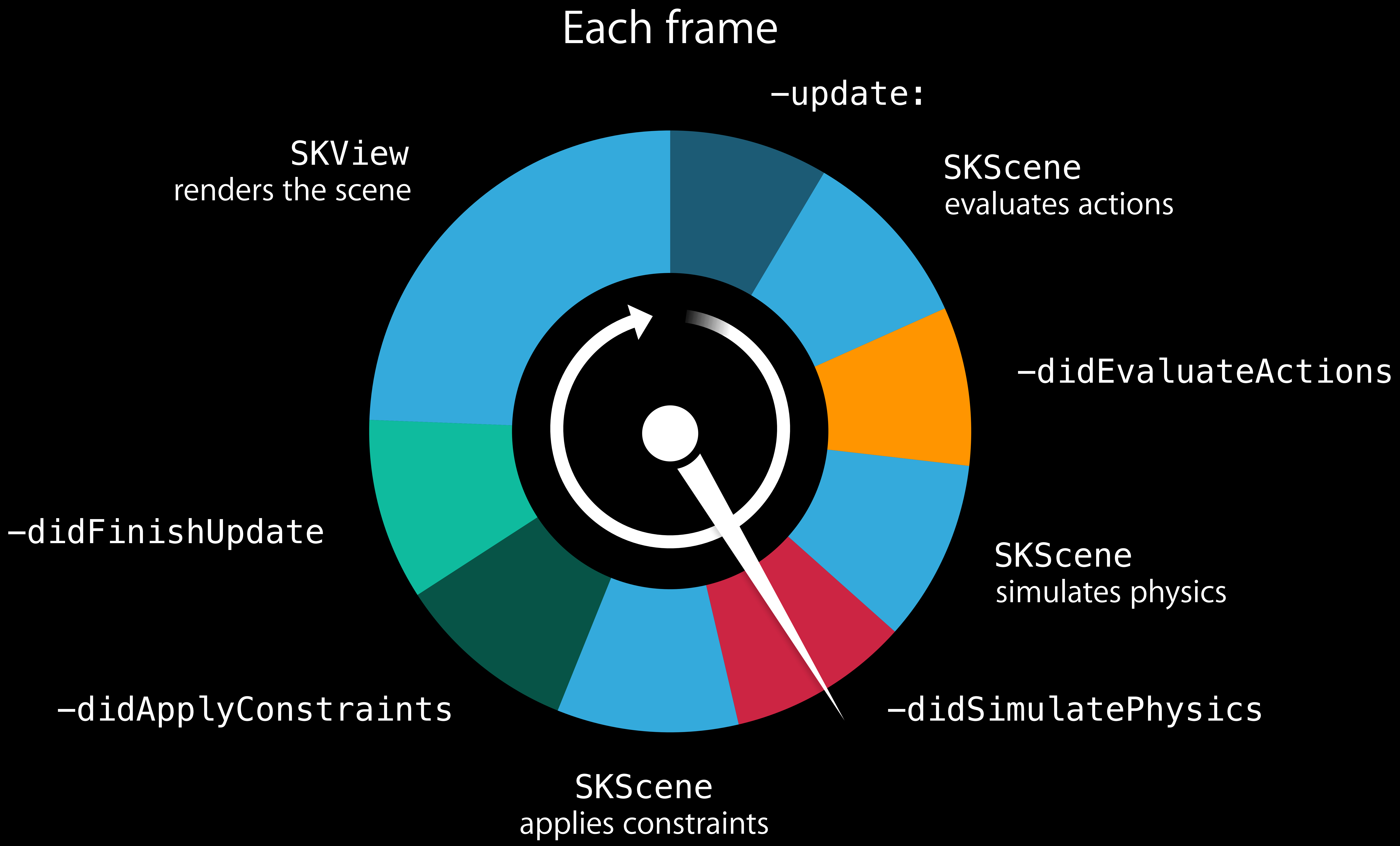

Each frame
-update:
SKScene
evaluates actions
-didEvaluateActions
SKScene
simulates physics
-didSimulatePhysics
SKScene
applies constraints
-didApplyConstraints
-didFinishUpdate
SKView
renders the scene

SKScene

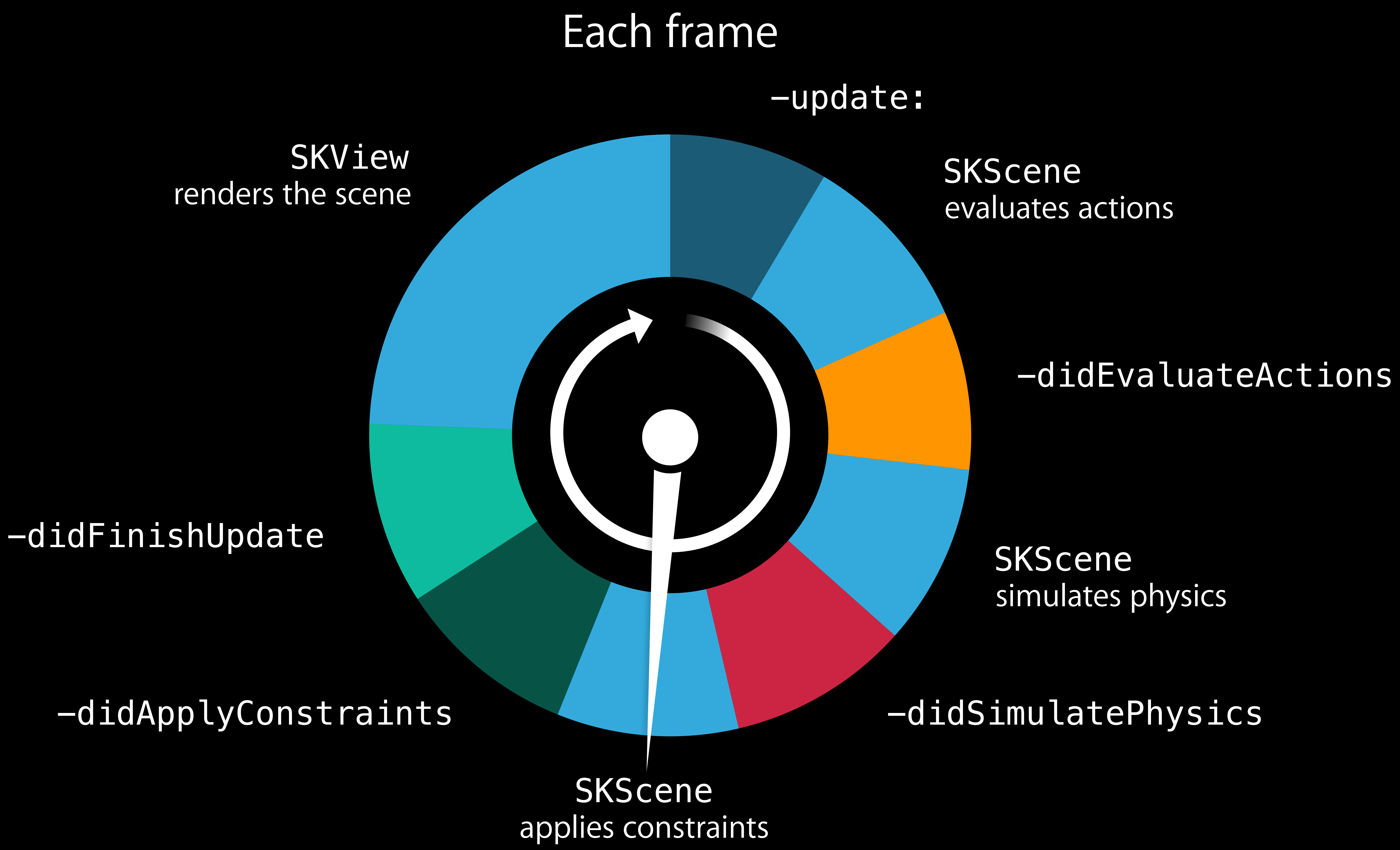


Each frame
-update:
SKScene
evaluates actions
-didEvaluateActions
SKScene
simulates physics
-didSimulatePhysics
SKScene
applies constraints
-didApplyConstraints
-didFinishUpdate
SKView
renders the scene

SKScene

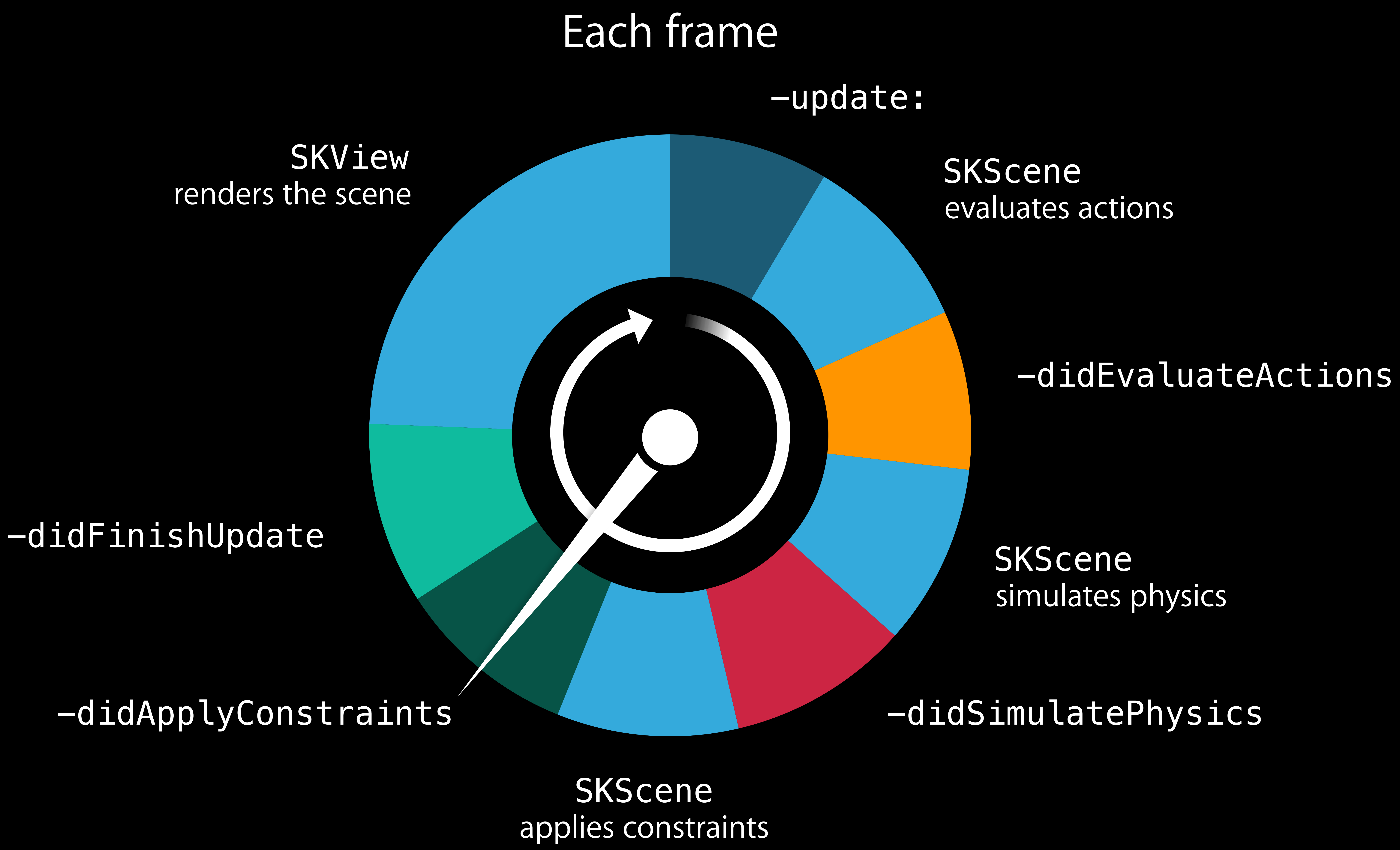

Each frame
-update:
SKScene
evaluates actions
-didEvaluateActions
SKScene
simulates physics
-didSimulatePhysics
SKScene
applies constraints
-didApplyConstraints
-didFinishUpdate
SKView
renders the scene

SKScene

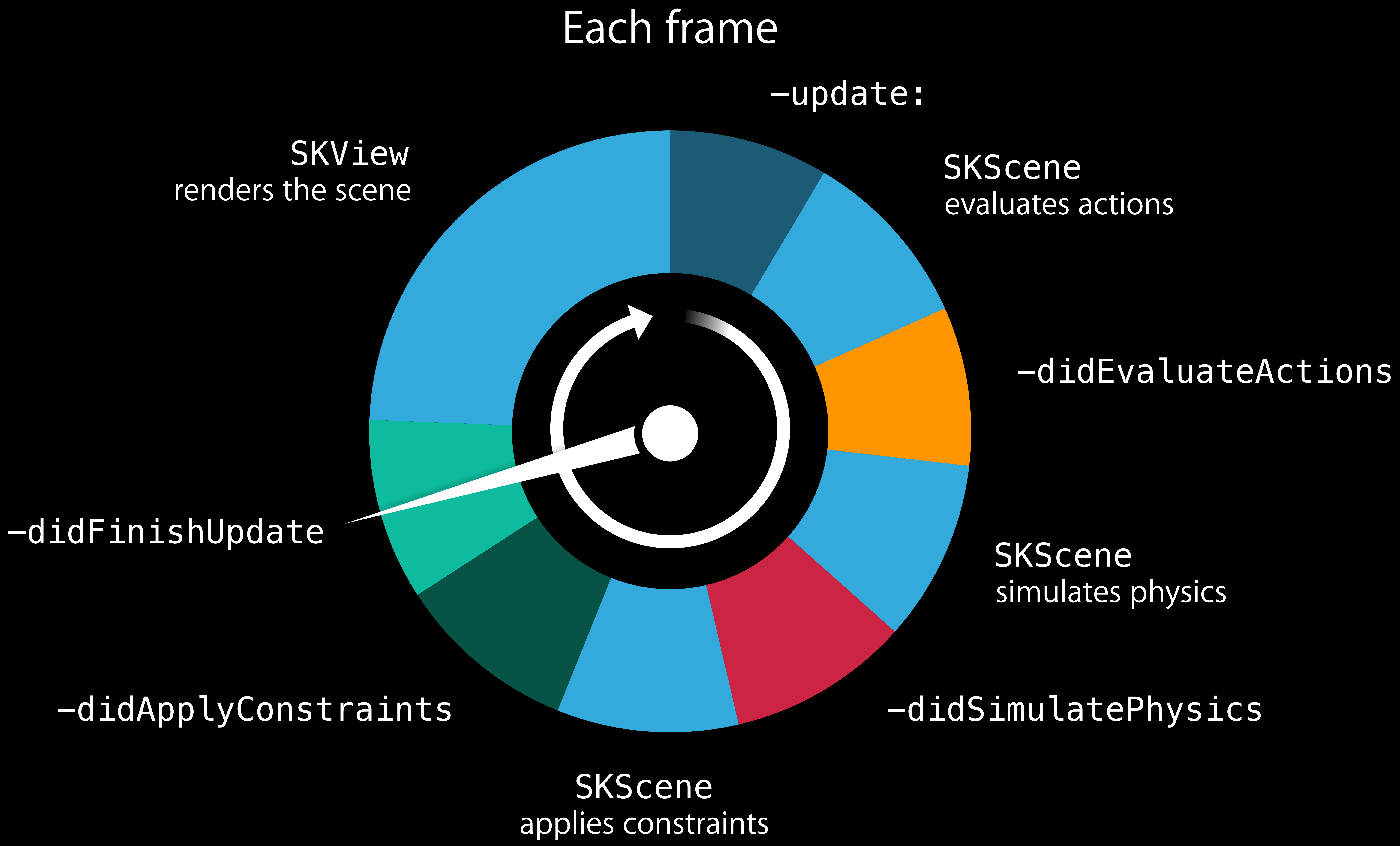


Each frame
-update:
SKScene
evaluates actions
-didEvaluateActions
SKScene
simulates physics
-didSimulatePhysics
SKScene
applies constraints
-didApplyConstraints
-didFinishUpdate
SKView
renders the scene

SKScene

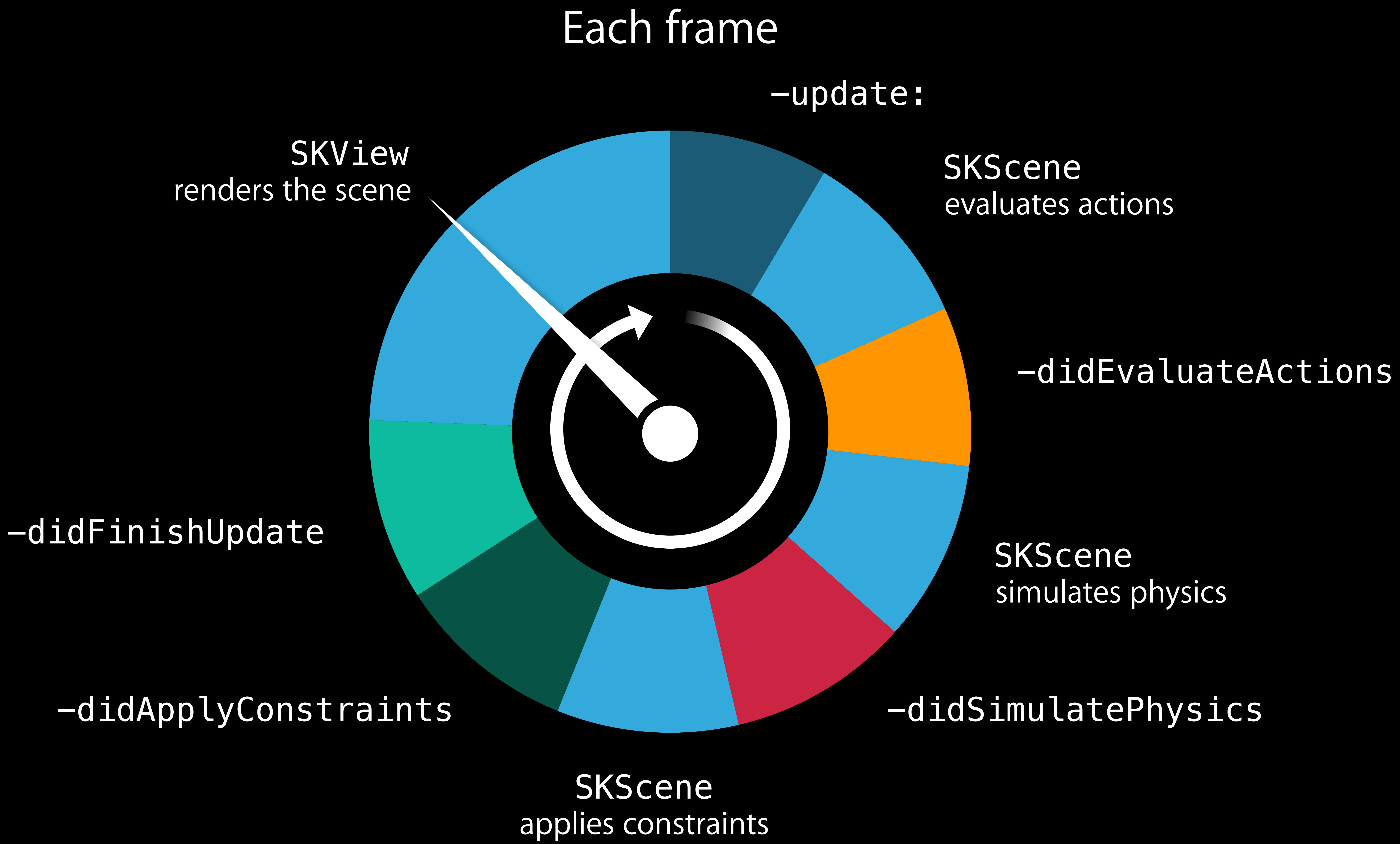

Each frame
-update:
SKScene
evaluates actions
-didEvaluateActions
SKScene
simulates physics
-didSimulatePhysics
SKScene
applies constraints
-didApplyConstraints
-didFinishUpdate
SKView
renders the scene

SKScene



Each frame
-update:
SKScene
evaluates actions
-didEvaluateActions
SKScene
simulates physics
-didSimulatePhysics
SKScene
applies constraints
-didApplyConstraints
-didFinishUpdate
SKView
renders the scene

SKScene


Each frame
-update:
SKScene
evaluates actions
-didEvaluateActions
SKScene
simulates physics
-didSimulatePhysics
SKScene
applies constraints
-didApplyConstraints
-didFinishUpdate
SKView
renders the scene

SKTexture

Mutable texture

SKMutableTexture

Create from data and can be modified efficiently

Provide code block to access raw pixel data

```
[tex modifyPixelDataWithBlock:^(void *pixelData,
                                 size_t lengthInBytes) {
   ...
}];
```

SKTexture

Mutable texture

SKMutableTexture

Create from data and can be modified efficiently

Provide code block to access raw pixel data

```
[tex modifyPixelDataWithBlock:^(void *pixelData,
                                size_t lengthInBytes) {
   ...
}];
```

SKTexture

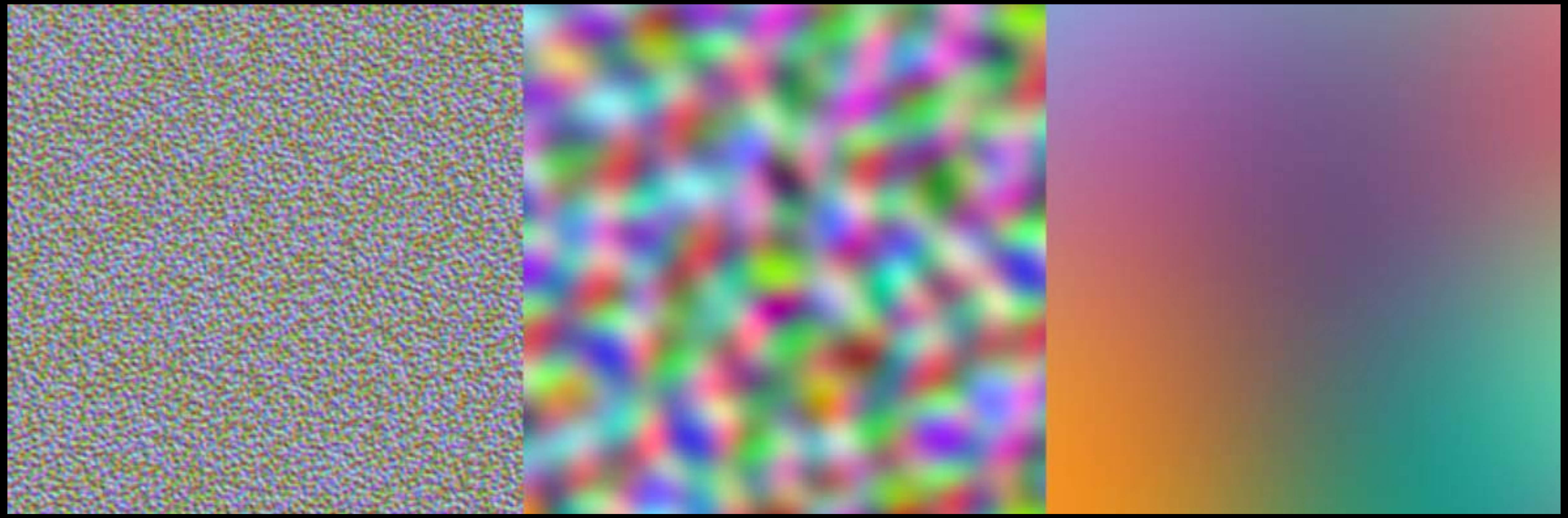
Noise texture

Generates coherent noise, or noise vector on a sphere

Supports grayscale and color output

Controls noise texture smoothness

```
texture = [SKTexture textureNoiseWithSmoothness:0
                                           size:CGSizeMake(s, s)
                                      grayscale:NO];
```



SKShapeNode

Convenient constructors for common shapes

- Rectangle, circle, ellipse, and spline

Simple joints for non-continuous shapes

Set texture and shaders on the shape’s stroke and fill

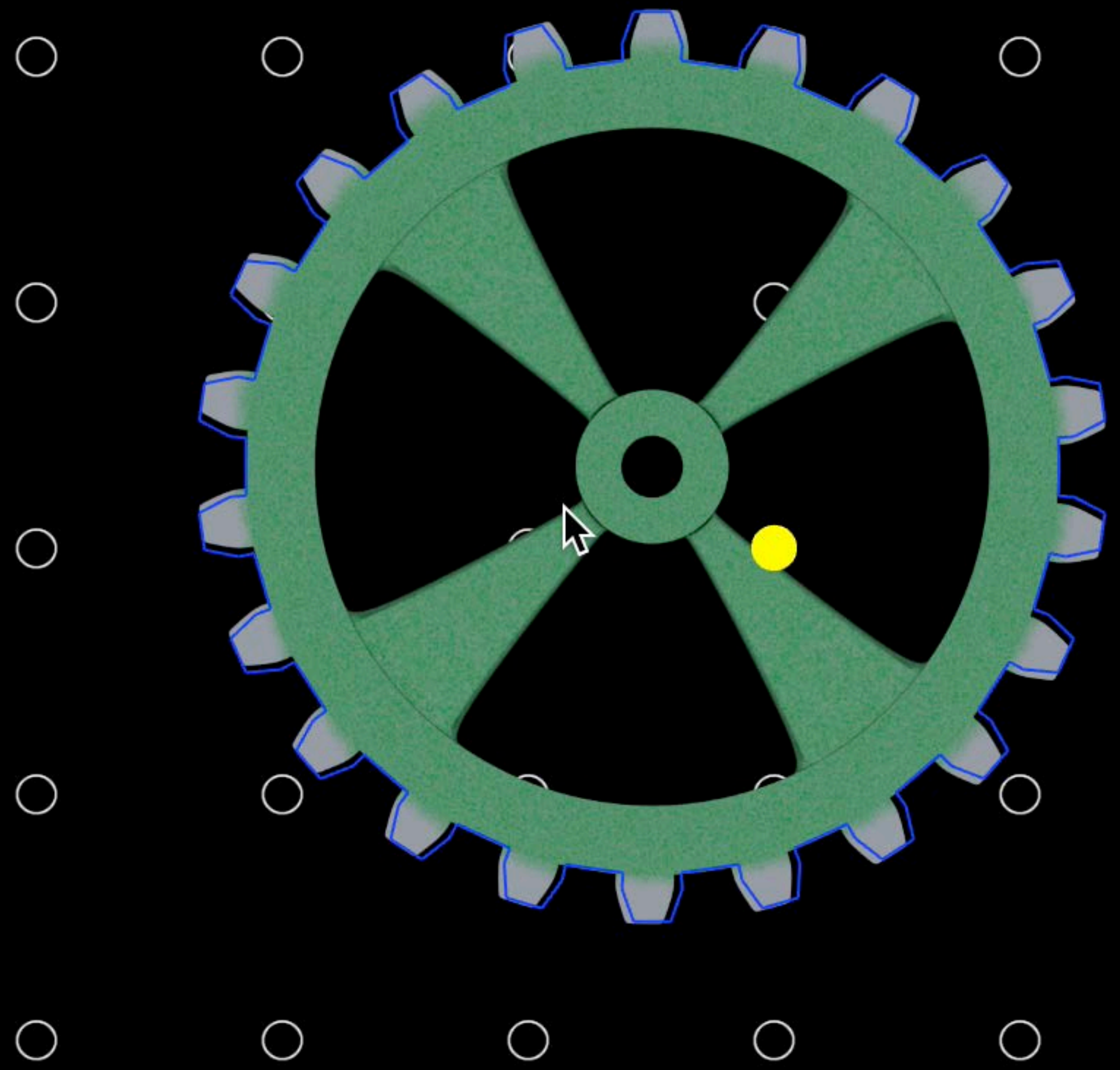
Interacting with physics

- path property

Physics Updates

Create pin joint

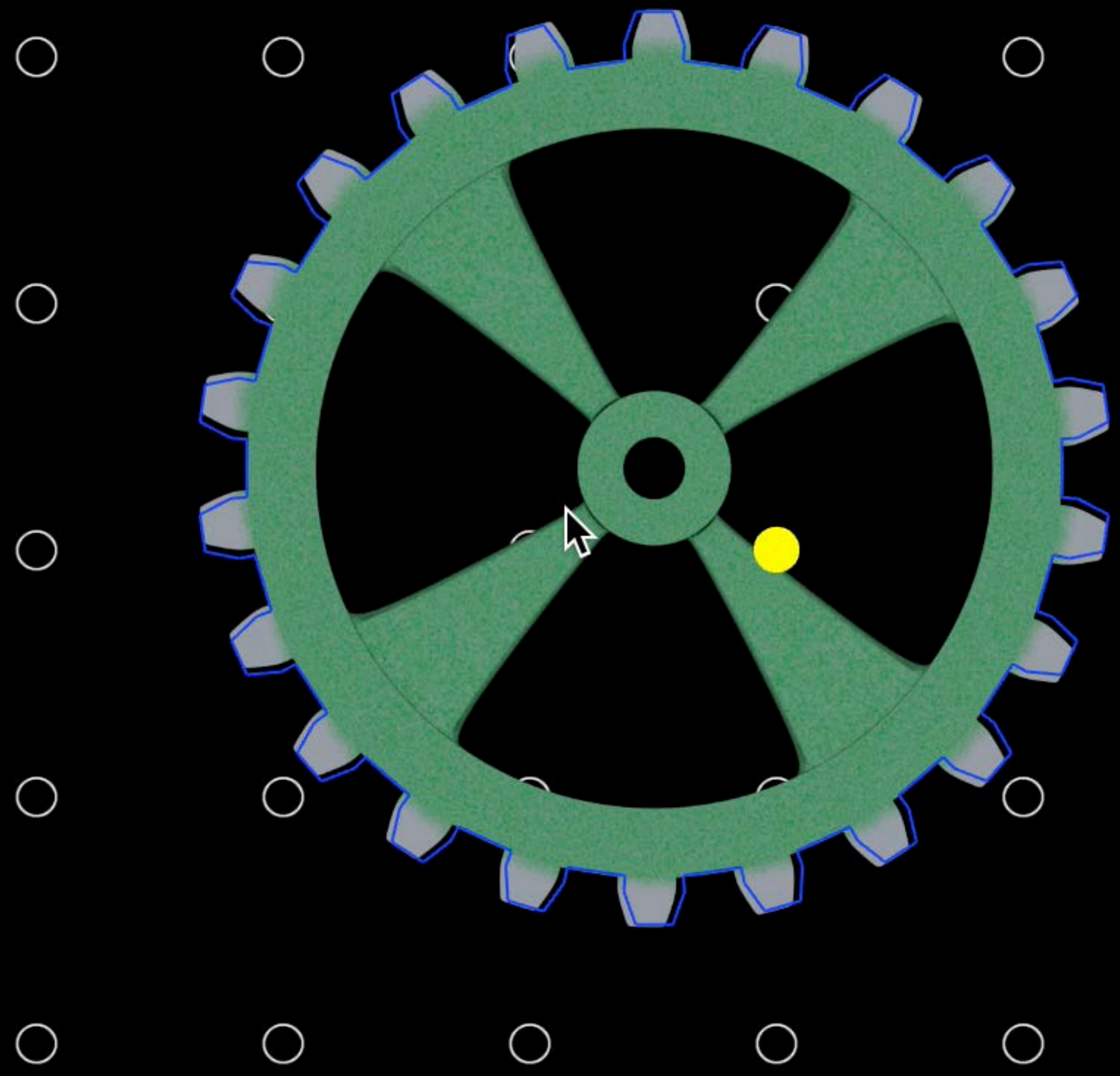
```
bigGear.physicsBody.pinned = YES;
```



Physics Updates

Create pin joint

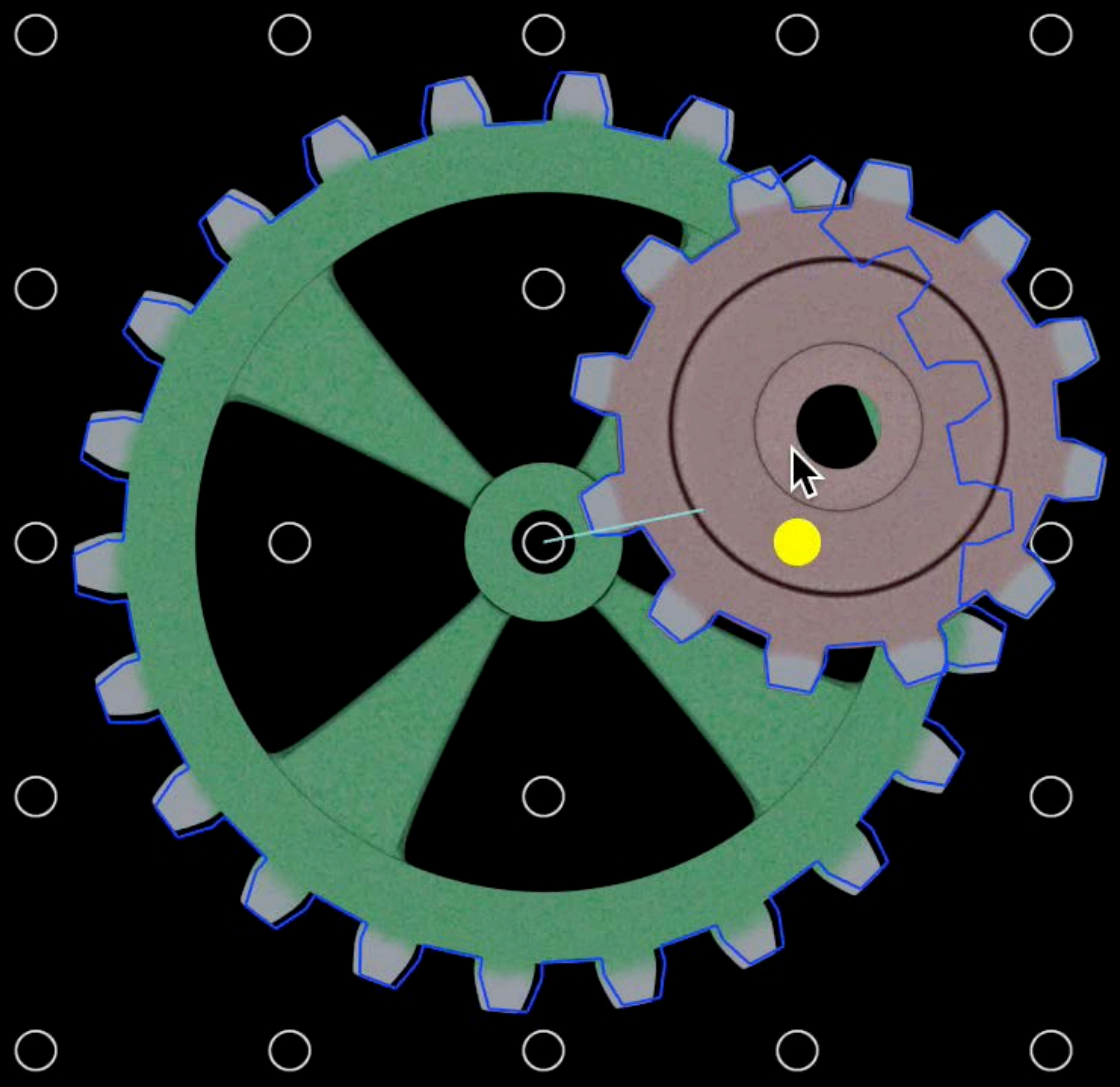
```
bigGear.physicsBody.pinned = YES;
```



Physics Update

Create weld joint

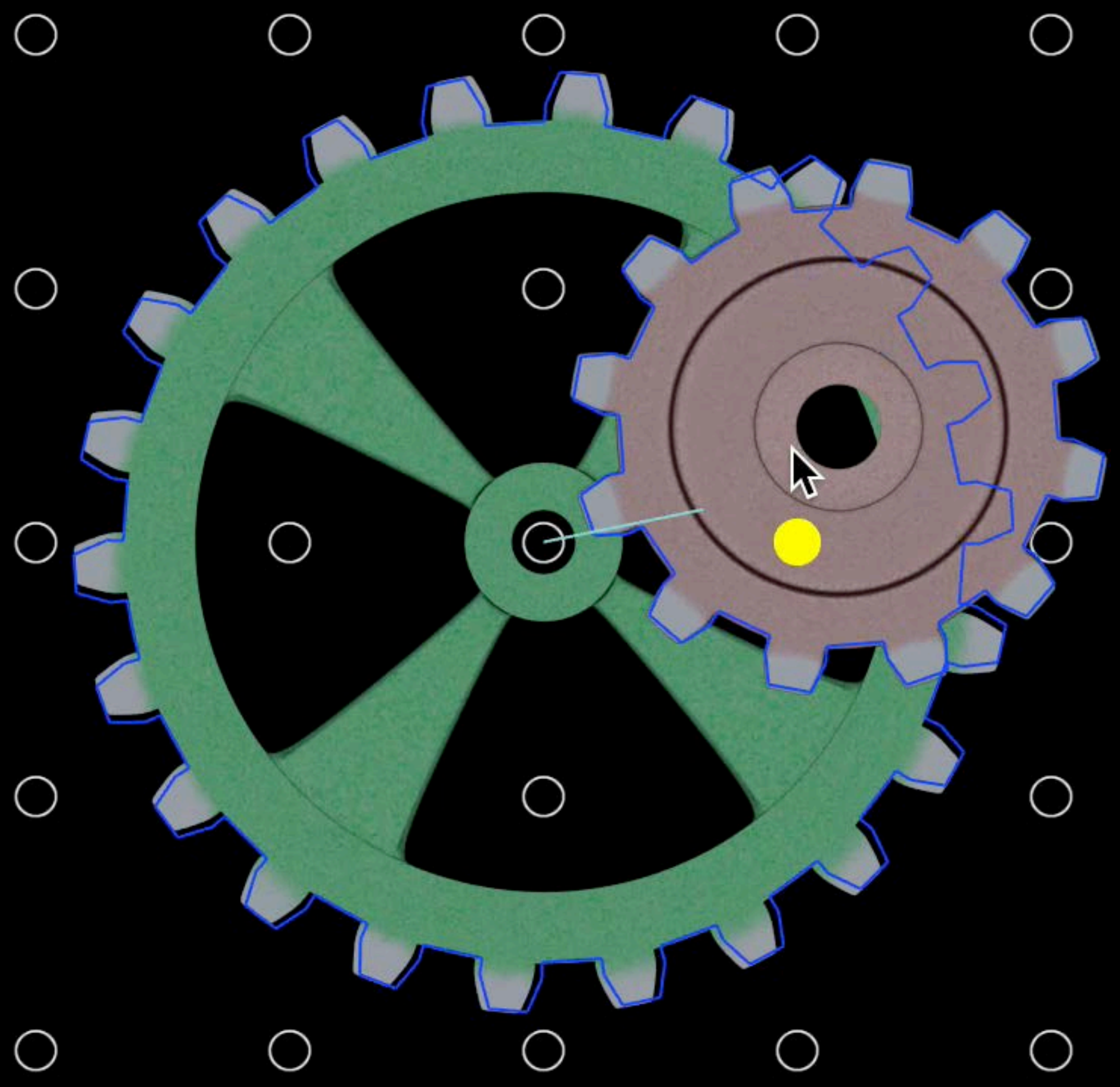
```
smallGear.physicsBody.pinned = YES;
smallGear.physicsBody.allowsRotation = NO;
```



Physics Update

Create weld joint

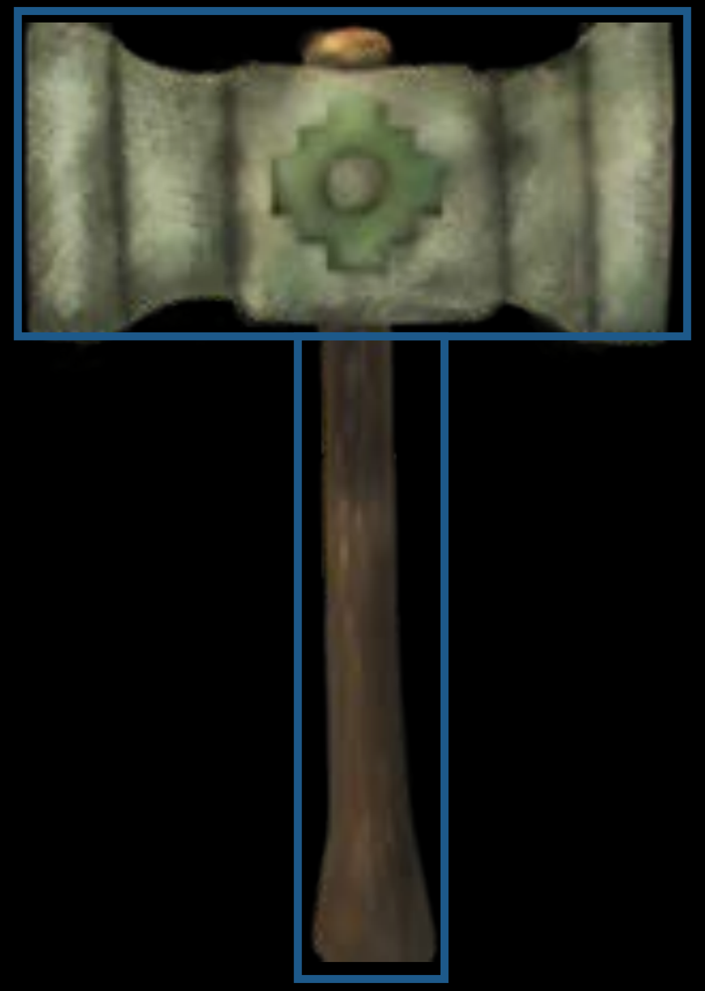
```
smallGear.physicsBody.pinned = YES;
smallGear.physicsBody.allowsRotation = NO;
```



Physics Update

Create compound bodies

```
+ (SKPhysicsBody *)bodyWithBodies:(NSArray *)bodies;
```



Texture Atlas

Generation

Supports SpriteKit and SceneKit

Supports Retina and non-Retina resolution

Supports 16-bit and 32-bit formats

- RGBA8888, RGBA4444, RGBA565, and RGBA5551

Supports up to 4096x4096 resolution

Texture Atlas

Generation

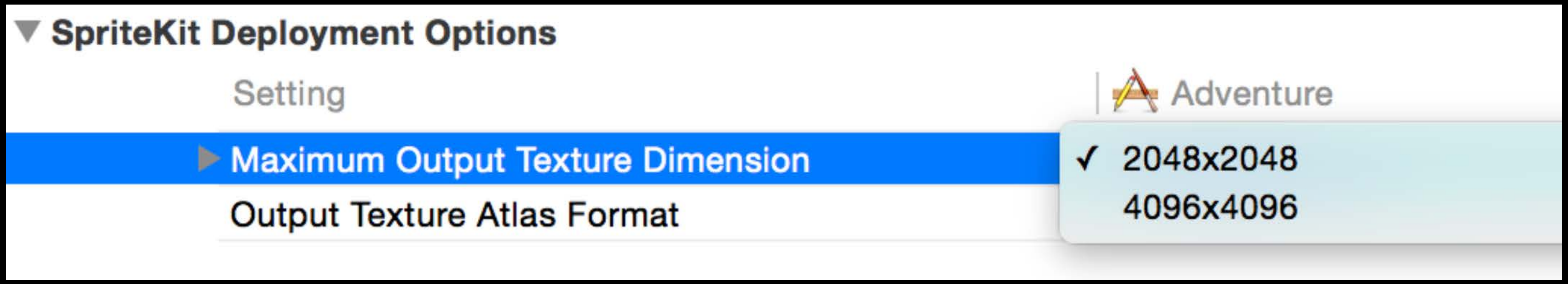
Supports SpriteKit and SceneKit

Supports Retina and non-Retina resolution

Supports 16-bit and 32-bit formats

- RGBA8888, RGBA4444, RGBA565, and RGBA5551

Supports up to 4096x4096 resolution



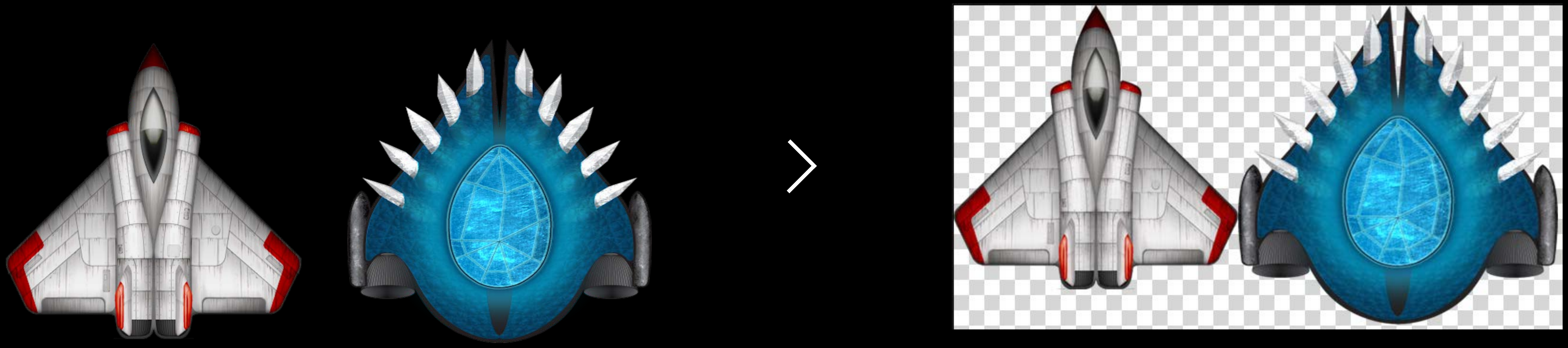
▼ SpriteKit Deployment Options

Setting	Adventure
▶ Maximum Output Texture Dimension	✓ 2048x2048 4096x4096
Output Texture Atlas Format	

Texture Atlas

Runtime generation

```
SKTextureAtlas *atlas = [SKTextureAtlas atlasWithDictionary:@{
    @“ship.png”  : image1,
    @“alien.png” : image2
}];
```



Summary

More Information

Allan Schaffer
Graphics and Game Technologies Evangelist
aschaffer@apple.com

Filip Iliescu
Graphics and Game Technologies Evangelist
filiescu@apple.com

Documentation
SpriteKit Programming Guide
http://developer.apple.com/library

Apple Developer Forums
http://devforums.apple.com

Related Sessions

Best Practices for Building SpriteKit Games	Pacific Heights	Wednesday 3:15PM
What's New in SceneKit	Pacific Heights	Thursday 10:15AM
Building a Game with SceneKit	Pacific Heights	Thursday 11:30AM

Labs

● SpriteKit Lab	Lab B	Thursday 12:45PM

WWDC14