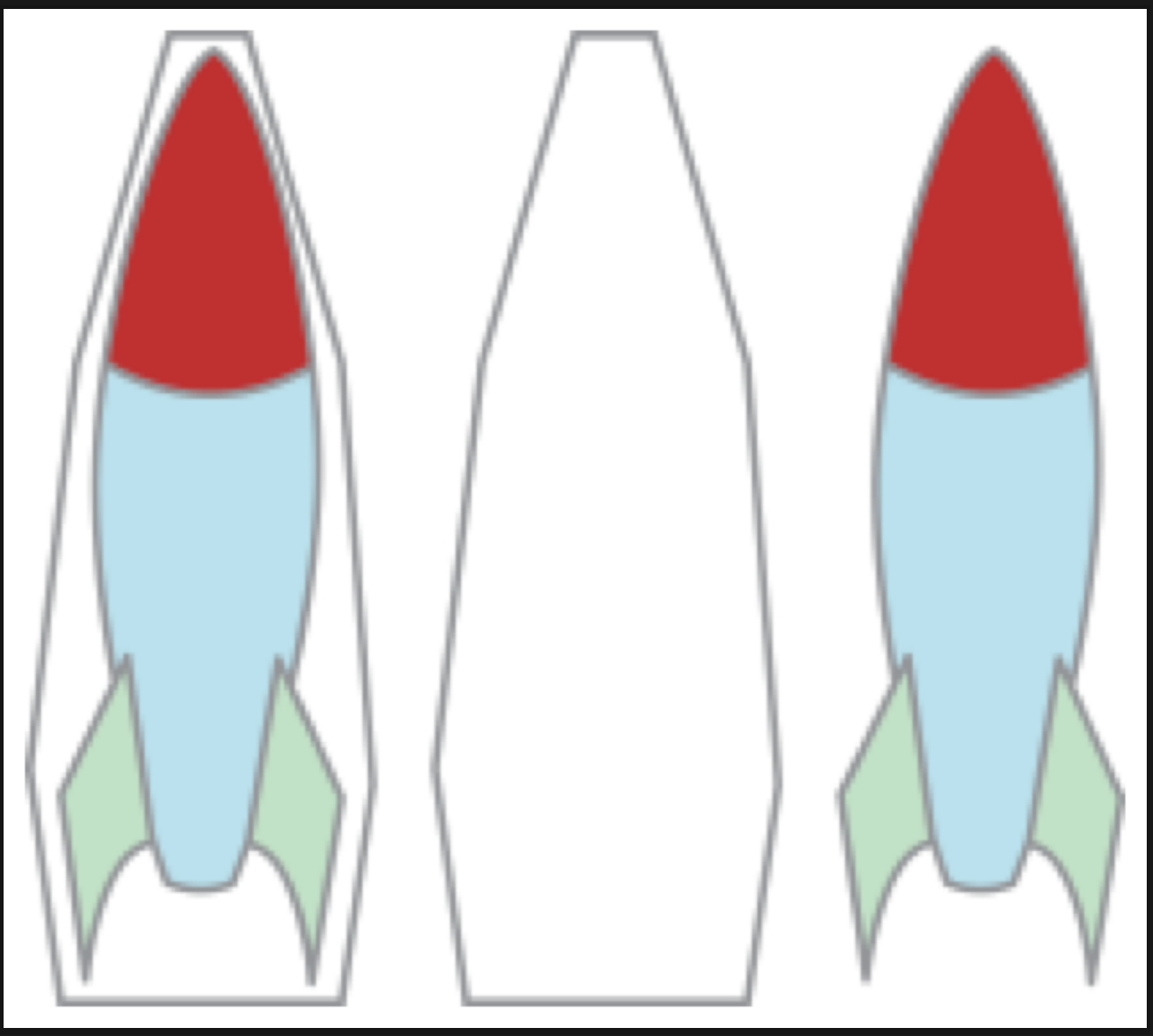
Shaping a Physics Body to Match a Node’s Graphics

Shape a physics body to your graphics for the right blend of collision accuracy and performance.

Overview

In most cases, a physics body should have a size and shape that closely approximates the visual representation of the corresponding node. For example, the rocket shown below has a narrow shape that is not well represented by either a circle or a rectangle. A convex polygon shape is chosen and fitted to match the sprite’s artwork.



Shape a Physics Body Using a Texture's Alpha Channel

If you do not want to create your own shapes, you can use SpriteKit to create a shape for you based on the sprite's texture.

```
let sprite = SKSpriteNode(imageNamed: "Spaceship")
sprite.physicsBody = SKPhysicsBody(texture: sprite.texture!,
                                   size: sprite.texture!.size())
```

Choose a Simple Geometric Physics Body Shape

When choosing a shape for your physics body, do not be overly precise. More complex shapes require more work to be properly simulated. For volume-based bodies, use the following guidelines:

>> A circle is the most efficient shape (init(circleOfRadius:))

>> A path-based polygon is the least efficient shape, and the computational work scales with the complexity of the polygon (init(polygonFrom:))

Use Edge-Based Physics Bodies Only When Needed

An edge-based body is more expensive to compute than a volume-based body. This is because the bodies it interacts with can potentially be on either side of an open edge or on the inside or outside of a closed shape. Use these guidelines:

>> Lines and rectangles are the most efficient edge-based bodies (init(edgeFrom:to:) and init(edgeLoopFrom:))

>> Edge loops and edge chains are the most expensive edge-based bodies, and the computational work scales with the complexity of the path (init(edgeLoopFrom:) and init(edgeChainFrom:)).