Além de jogar, eu faço jogos!

Comitê para Democratização da Informática

Uma parceria de sucesso entre o CPDI e o Instituto Cooperforte

Professora Juliana Oliveira

Pauta – Aula 13

O que vamos aprender hoje

01

Continuação da biblioteca de física.

Física

A biblioteca de física em Lua trata as propriedades do corpo físico. Qualquer objeto de exibição padrão pode ser tornado físico e interagir automaticamente com outros objetos na simulação.

Para começar a usar física, precisamos "chamar" a biblioteca.

Corpos adormecidos

Corpos físicos não envolvidos em uma colisão "dormem" após alguns segundos. Isso reduz a sobrecarga de desempenho, mas isso pode ser substituído em casos que se faz necessário a física estar sempre ativa.

Opções de simulação

O mecanismo de física apresenta várias opções de simulação global. Podem ser usados para ajustar a simulação física além do comportamento padrão.

physics.setGravity ()

Define os componentes x e y do vetor de gravidade global, em unidades de m/s². O padrão é simular a gravidade padrão da Terra, (0, 6)

physics.setScale ()

Define a proporção interna de pixels por metro que é usada na conversão entre as coordenadas na tela e as coordenadas físicas simuladas. Deve ser feito apenas uma vez, utilizado no caso de os objetos parecerem lentos ou reagirem muito lentamente. O valor de escala padrão é 30.

physics.setDrawMode()

Define um dos 3 modos de renderização possíveis para o mecanismo de física. Ele é executado em dispositivos porém utilizamos apenas no simulador para verificar o comportamento do mecanismo de física.

physics.setScale ()

Os dados físicos são exibidos usando gráficos vetoriais coloridos que refletem diferentes tipos e atributos de objetos.

- Laranja - corpos físicos dinâmicos.
- Azul escuro - corpos de física cinemática
- Verde - Corpos estáticos e não móveis.
- Cinza - Corpo que está "dormindo" devido a falta de atividade.
- Azul claro - Juntas físicas.

physics.setPositionIterations ()

Define a precisão dos cálculos de posição do mecanismo. O valor padrão é 3, aumentar esse número causará menos imprecisões momentâneas, como objetos sobrepostos, mas aumetará a sobrecarga computacional. O padrão deve funcionar para a maioria dos casos.

physics.setContinuous()

Controla se a física contínua está habilitada. Por padrão, a detecção de colisão é contínua, o que impede que os objetos "atuem em túnel". Se fosse desligado, um objeto que se move com rapidez suficiente poderia passar por uma parede fina.

physics.setAverageCollisionPositions ()

Permite a média de todos os pontos de contato em uma colisão entre corpos físicos. Essa configuração é útil para relatar os pontos de colisão como um ponto médio.

physics.setReportCollisionsInContentCoordinates()

Define a origem do conteúdo como o ponto de referência nos eventos físicos de colisão.

physics.setDebugErrorsEnabled()

Permite que erros físicos extras sejam detectados pelo simulador. O padrão é true (ativado).

physics.setTimeStep()

Especifica uma simulação física baseada em quadro (aproximada) ou uma simulação baseada em tempo.

physics.setMKS()

Define o valor MKS (metros, quilogramas e segundos) da simulação física para chaves específicas. Isso é estritamente para fins avançados.

Corpos de física

O mundo da física é baseado nas interações dos corpos físicos. Corona trata esses corpos físicos como uma extensão de seus objetos gráficos. Atributos de leitura/gravação de objeto padrão como x, y e rotation devem continuar a funcionar normalmente em corpos físicos. No entanto, se o objeto bodyTypefor dinâmico, o mecanismo de física pode "revidar" contra suas tentativas de mover o objeto manualmente, pois pode estar sob o efeito constante da gravidade ou de outras forças.

Criando corpos

Todos os corpos são criados com a função physics.addBody(). Isso permite que qualquer objeto de exibição seja transformado em um objeto físico.

Propriedades físicas

Todos os corpos físicos possuem 3 propriedades principais e cada uma pode ser definida como um par chave-valor na tabela de propriedades. Essas propriedades são opcionais e os valores padrão serão aplicados a menos que seja especificado de outra forma.

Densidade (density)

É multiplicado pela área da forma do corpo para determinar sua massa. O parâmetro é baseado em um padrão de 1.0 para água, os materiais mais leves que a água tem uma densidade abaixo e os mais pesados acima. Podemos ajustar os valores de densidade conforme necessidade do objeto no jogo. O comportamento geral também vai depender das configurações de gravidade e escala.

Atrito (friction)

Pode ser qualquer valor não negativo. 0 significa sem atrito e 1.0 atrito bem forte. O padrão é 0.

Quicar (bounce)

Determina quanto da velocidade do objeto retorna após uma colisão. Valores maiores que 0.3 são razoavelmente saltantes, o valor padrão é 0.2.

Tipos de corpo

Os corpos físicos podem ser de 3 tipos: dinâmicos, estáticos ou cinemáticos.

Dinâmico (dynamic)

Corpos dinâmicos podem ser movidos manualmente em código, mas também se movem de acordo com forças como gravidade ou forças de colisão. É o tipo de corpo padrão para objetos físicos. **Podem colidir com qualquer tipo de corpo.**

Estático (Static)

Não se movem sob simulação e se comportam como se tivessem massa infinita. Podem ser movidos manualmente mas não aceitam aplicação de velocidade. **Colidem apenas com corpos dinâmicos.**

Cinemático (Kinematic)

Se movem apenas de acordo com sua velocidade. Não respondem a forças como gravidade. **Colidem apenas com corpos dinâmicos.**

Corpos retangulares

Por padrão, o comando de adicionar corpo aplicará um corpo retangular que circunda todas as bordas da imagem ou objeto.

Corpos circulares

São criados adicionando o radius à tabela de propriedades. Isso funciona bem para bolas, pedras e outros objetos que podem ser considerados aproximadamente redondos.

Corpos poligonais

Poden ser criados usando o parâmetro shape. É uma tabela de pares de coordenadas x e y onde cada par define um ponto de vértice para a forma.

São especificadas em relação ao centro do objeto.

Os pontos devem ser definidos no sentido horário.

As formas podem ter no máximo 8 pontos de vértice.

Corpos retangulares deslocados/angulares

Para criar um corpo retangular que não abranja toda a largura e altura do objeto, podemos definir o parâmetro box e especificar os pares de valores dentro dele. Esse tipo de corpo também pode ser deslocado do centro do objeto ou definido para um ângulo diferente de 0.

- `halfWidth` — Metade da largura do corpo. Esta propriedade é obrigatória.
- `halfHeight` - Metade da altura do corpo. Esta propriedade é obrigatória.
- `x` — O deslocamento **x** (±) do centro do objeto de exibição. Essa propriedade é opcional e o padrão é `0`.
- `y` — O deslocamento **y** (±) do centro do objeto de exibição. Essa propriedade é opcional e o padrão é `0`.
- `angle` — O ângulo (rotação) do corpo. Essa propriedade é opcional e o padrão é `0`.

Corpos de contorno

O Corona trás a opção de delinear o objeto de exibição caso não deseje definir uma forma de corpo específica. Corpos contornados tem menos restrições do que corpos poligonais por exemplo, uma forma de corpo contornada pode ser convexa ou côncava.

Sensores

Qualquer corpo pode ser transformado em um sensor. Sensores não interagem fisicamente com outros corpos, mas produzem eventos de colisão quando outros corpos passam por eles. É útil se precisarmos detectar quando o objeto colide com uma região específica, mas não causa nenhuma reação de força como resultado.

Propriedades do corpo

object.bodyType

É um valor para o tipo de corpo físico. Incluem dynamic, static ou kinematic.

object.isAwake

É o valor booleano para determinar se o corpo tem permissão para dormir. Manter um corpo acordado tem uma sobrecarga de desempenho maior e geralmente não é necessário porque uma colisão com outro corpo o acordará automaticamente.

object.isBodyActive

usado para definir ou obter o estado ativo atual do corpo. Corpos inativos não são destruídos, mas são removidos da simulação física e deixam de interagir com outros corpos. Não podemos alterar o estado ativo do corpo durante um evento de colisão, portanto, devemos enfileirar esse evento após um pequeno e imperceptível atraso.

object.isSensor

Define a propriedade interna de sensor em todos os elementos do corpo. Como atua em todos os elementos do corpo, substitui incondicionalmente qualquer configuração de sensor nos elementos individuais.

object.isFixedRotation

É o valor booleano para determinar se a rotação do corpo deve ser travada, mesmo se o corpo estiver sujeito a forças descentralizadas.

object.gravityScale

pode ser usado para ajustar o efeito da gravidade em um corpo específico. Por exemplo, configurá-lo para 0 fazer o corpo flutuar, mesmo que outros objetos na simulação estejam sujeitos à gravidade normal. O valor padrão é 1.0, significando gravidade normal.

object.angularVelocity

Valor numérico da velocidade angular (rotacional) atual, em graus por segundo.

object.angularDamping

Valor numérico de quanto a rotação do corpo deve ser amortecida — como a rapidez com que o objeto giratório desacelerará até um ponto final no sentido rotacional (não linear). O padrão é 0, o que significa que o corpo irá girar na mesma velocidade indefinidamente.

object.linearDamping

Valor numérico de quanto o movimento linear do corpo é amortecido — como a rapidez com que o objeto desacelerará até um ponto final no sentido linear (não rotacional). O padrão é 0, o que significa que o corpo se moverá na mesma velocidade indefinidamente.

object.isBullet

Valor booleano para se o corpo deve ser tratado como um "bala" em relação à detecção de colisão. As balas estão sujeitas a detecção de colisão contínua em vez de detecção periódica. Isso é computacionalmente mais caro, mas pode impedir que objetos em movimento rápido passem por barreiras sólidas.

Métodos do corpo

object:setLinearVelocity()

Define os componentes x e y para a velocidade linear do corpo, em pixels por segundo.

object:getLinearVelocity()

Retorna os componentes xey para a velocidade linear do corpo, em pixels por segundo. Esta função aproveita o fato de que Lua pode retornar vários valores, neste caso ambas as velocidades lineares.

object:applyForce()

Aplica os componentes xey especificados de uma força linear em um determinado ponto dentro das coordenadas mundiais. Se o ponto alvo for o centro de massa do corpo, ele tenderá a empurrar o corpo em linha reta. Se o ponto alvo estiver deslocado do centro, o corpo girará em torno de seu centro de massa. Para objetos simétricos, o centro de massa e o centro do objeto terão a mesma posição.

object:applyLinearImpulse()

semelhante a object:applyForce(), exceto que um impulso é uma única e momentânea sacudida de força. O impulso pode ser aplicado a qualquer ponto do corpo (seja o centro de massa ou um ponto de deslocamento).

object:applyTorque()

Aplica uma força rotacional ao corpo. Valores positivos resultarão em torque no sentido horário; valores negativos resultarão em torque no sentido anti-horário. O corpo irá girar em torno de seu centro de massa.

object:applyAngularImpulse()

Semelhante a object:applyTorque(), exceto que um impulso é uma única e momentânea sacudida de força.

object:resetMassData()

É útil se os dados de massa padrão para o corpo foram substituídos. Esta função redefine para a massa calculada a partir das formas.

Força e impulso

Uma pergunta comum é se aplicamos força ou impulso a um corpo. A diferença é que um impulso serve para simular um choque/chute imediato no corpo, enquanto força (e torque) é algo exercido ao longo do tempo. Portanto, para obter uma simulação de força/torque realista, você deve aplicá-la continuamente em cada ciclo de aplicação, enquanto desejar que a força continue.