Aula 7

ALÉM DE JOGAR,

EU FAÇO JOGOS!

Instituto COOPERFORTE

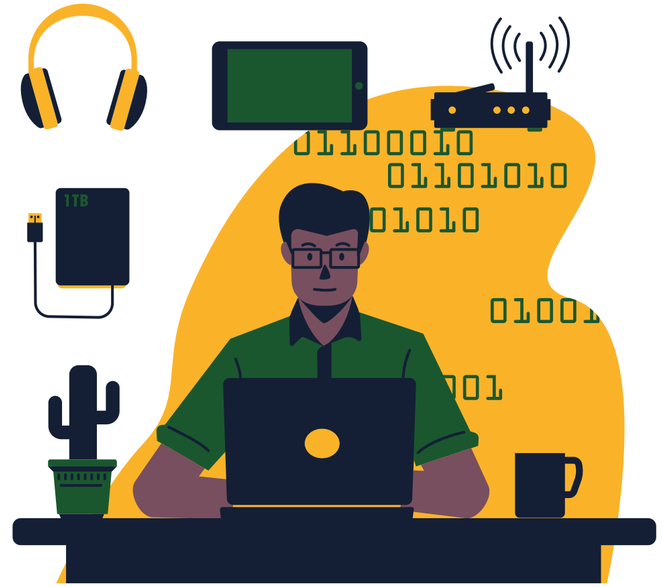
20 anos transformando realidades

cpdi

Comitê para Democratização da Informática

O que vamos aprender hoje?

- Colisões



Eventos de colisão

Os eventos de colisão do motor de física são expostos por meio do modelo padrão de ouvinte de eventos Corona com 3 tipos:

Colisão

Para detecção geral de colisão usamos o ouvinte de evento "collision". Ele inclui fases para "began" e "ended" que são os momentos de contato inicial e contato rompido. Se ele não for implementado o evento não será acionado.

Pré-colisão

O evento "preCollision" é acionado diretamente antes que os objetos comecem a interagir. Esses eventos são muito "ruidosos" e podem ser relatados muitas vezes por contato, afetando o desempenho. Só deve-se usar se realmente for necessária uma detecção pré-colisão. Também é melhor evitar usar esse evento no Runtime, sendo indicado apenas para ouvintes locais.

Pós-colisão

O "postCollision" é acionado diretamente após a interação dos objetos. É o único evento em que a força de colisão e o atrito são relatados. Assim como o preCollision, são eventos muito ruidosos e afetam o desempenho. Então as recomendações de uso são as mesmas do pré-colisão.

Lembrando que, em uma colisão entre dois objetos físicos, pelo menos um dos objetos deve ser dinâmico, pois este é o único tipo de corpo que colide com qualquer outro tipo.

Colisões

Alguns métodos não podem ser chamados durante um evento de colisão, são os seguintes:

- object.bodyType
- object:rotate()
- object.rotation
- object:translate()
- object.x
- object.y
- physics.addBody()
- physics.removeBody()
- physics.newJoint()
- physics.newParticleSystem()
- physics.setContinuous()
- physics.stop()

Regras de propagação

Os eventos de colisão possuem um modelo de propagação semelhante aos eventos de touch. Podemos usar isso para otimizar ainda mais o desempenho do jogo, limitando o número de eventos que serão criados. Por padrão, uma colisão entre dois objetos acionará um evento local para o primeiro objeto, depois outro para o segundo objeto e em seguida um global no objeto Runtime, supondo que todos tenham ouvintes ativos. No entanto, pode ser que precisemos apenas de algumas dessas informações.

Regras de propagação

Qualquer manipulador de eventos de colisão que retornar true interromperá a propagação dele, mesmo se houverem mais ouvintes que poderiam ter o recebido. Isso permite limitar o número de eventos que são criados. Embora eventos individuais não sejam muito caros, um grande número deles pode afetar o desempenho geral, portanto, limitar a propagação de eventos é uma boa prática.

Tratamento de colisões

As colisões são relatadas entre pares de objetos e podem ser detectadas localmente usando um ouvinte de objeto ou globalmente usando um ouvinte de tempo de execução.

Tratamento de colisões locais

O manuseio de colisão local é melhor utilizado em um cenário de colisão de um para muitos, por exemplo um objeto que pode colidir com vários inimigos, power-ups etc. Para manuseá-la, cada evento possui um self ID representando o objeto em si e o event.other que contém o ID do outro objeto envolvido na colisão.

Tratamento de colisões

As colisões são relatadas entre pares de objetos e podem ser detectadas localmente usando um ouvinte de objeto ou globalmente usando um ouvinte de tempo de execução.

Tratamento de colisões locais

O manuseio de colisão local é melhor utilizado em um cenário de colisão de um para muitos, por exemplo um objeto que pode colidir com vários inimigos, power-ups etc. Para manuseá-la, cada evento possui um self ID representando o objeto em si e o event.other que contém o ID do outro objeto envolvido na colisão.

Tratamento de colisões globais

O manuseio de colisão global é melhor utilizado em um cenário de muitos para muitos, por exemplo vários personagens heróis que podem colidir com vários inimigos. Para tratá-la, cada evento inclui event.object1 e event.object2 que indica as referências aos dois objetos envolvidos. Podemos armazenar o nome de cada objeto em formato de string e recuperar durante o evento de colisão.

Ao detectar colisões com o método global, não tem como determinar qual é o primeiro e o segundo objeto envolvido na colisão. Ou seja, event.object1 e 2 podem ser qualquer um dos objetos físicos do projeto. Assim, se estivermos comparando os dois objetos em uma instrução condicional, talvez seja necessário construir uma cláusula multicondicional para detectar ambas as possibilidades.