Aula 23 – Movimentação de Personagem e Input

Bem-vindo à Aula 23 do nosso Curso de Desenvolvimento de Jogos 3D! Se você já passou horas imerso em mundos virtuais, controlando heróis e explorando paisagens fantásticas, sabe que a sensação de poder mover um personagem de forma fluida e responsiva é a espinha dorsal de qualquer boa experiência de jogo. É essa conexão direta entre o jogador e o avatar na tela que transforma um conjunto de pixels em uma aventura memorável.

Nesta aula, vamos desvendar a magia por trás dessa interação fundamental. Entenderemos como os motores de jogo traduzem seus comandos do teclado, mouse ou gamepad em ações concretas do personagem, desde um simples passo até um salto acrobático. É um conhecimento essencial, pois a qualidade da movimentação e do sistema de input pode ser o diferencial entre um jogo que cativa e um que frustra.

Ao final desta jornada, você será capaz de compreender e aplicar os conceitos do Character Movement Component, configurar sistemas de Input eficientes usando Action e Axis Mappings, e construir a lógica de um personagem em terceira pessoa, incluindo seu controle e a dinâmica da câmera. Prepare-se para dar vida aos seus próprios avatares, garantindo que cada movimento seja tão intuitivo quanto gratificante. Vamos mergulhar nos detalhes que transformam uma ideia em uma experiência jogável e envolvente, conectando o que você já sabe sobre design de jogos com as ferramentas práticas de implementação.

O Coração da Animação: Character Movement Component

Motor do Personagem

Gerencia toda a movimentação física e interação com o ambiente

Sistema de Colisões

Garante que o personagem não atravesse paredes ou objetos

Física Realista

Aplica gravidade, atrito e outras forças naturais

Imagine que você está dirigindo um carro. Não basta apenas pisar no acelerador; você precisa de um motor, um sistema de direção, freios e uma suspensão que lide com as irregularidades do terreno. No mundo dos jogos 3D, nossos personagens são como esses carros complexos, e o "motor" que gerencia sua movimentação é frequentemente encapsulado em um componente especializado. Sem ele, seu personagem seria apenas um modelo estático, incapaz de interagir de forma crível com o ambiente.

Este componente, conhecido como Character Movement Component (no Unreal Engine) ou Character Controller (no Unity), é a peça central que orquestra a dança entre a física do mundo, as colisões com obstáculos e as animações do seu personagem. Ele não apenas move o modelo, mas também lida com a gravidade, o atrito, a capacidade de pular, agachar, nadar e até mesmo escalar. É ele quem garante que seu personagem não atravesse paredes ou flutue de forma irrealista, mantendo a imersão do jogador.

Pense nele como o cérebro e o sistema nervoso do seu personagem. Ele recebe os impulsos (seus comandos de input), processa-os e coordena os músculos (as animações) para que o corpo (o modelo 3D) se mova de forma coesa e realista dentro das regras do mundo do jogo. É uma abstração poderosa que simplifica enormemente a tarefa de criar movimentação complexa, permitindo que você se concentre na lógica de jogo em vez de reinventar a roda da física a cada novo projeto.

Detalhando o Character Movement Component

Agora que entendemos a função central do Character Movement Component, vamos explorar um pouco mais suas capacidades. Este componente é uma verdadeira caixa de ferramentas, repleta de parâmetros que permitem ajustar finamente o comportamento do seu personagem. Não se trata apenas de "andar para frente", mas de definir *como* ele anda para frente, *quão rápido* ele corre, *quão alto* ele pula e *com que agilidade* ele se vira.

Parâmetros Essenciais

- **Velocidade máxima** de caminhada e corrida
- **Aceleração e desaceleração** do movimento
- **Força do pulo** e altura máxima
- **Influência da gravidade** sobre o personagem
- **Modos de movimento** (andar, voar, nadar)

Entre os parâmetros mais comuns, encontramos a velocidade máxima de caminhada e corrida, a aceleração e desaceleração, a força do pulo, a influência da gravidade e até mesmo a capacidade de se mover em diferentes modos (como andar, voar ou nadar). Ajustar esses valores é como calibrar um instrumento musical: pequenas mudanças podem ter um impacto gigantesco na "sensação" do controle do personagem. Um personagem muito lento pode parecer pesado e desajeitado, enquanto um muito rápido pode ser difícil de controlar com precisão.

A beleza desse componente reside na sua flexibilidade. Você pode, por exemplo, criar um personagem ágil e leve para um jogo de plataforma, ou um guerreiro pesado e lento para um RPG de ação, simplesmente ajustando alguns sliders. Essa capacidade de personalização é crucial para que a movimentação do personagem se alinhe perfeitamente com a identidade e o gênero do seu jogo, impactando diretamente a experiência do jogador e a imersão no mundo que você está construindo.

A Ponte entre Jogador e Jogo: O Sistema de Input

O Tradutor Universal

O sistema de Input é responsável por "ouvir" os dispositivos de entrada e traduzir esses sinais em eventos que o jogo pode entender e reagir. Ele permite que você mapeie comandos específicos – como "pular", "atirar" ou "mover para frente" – para diferentes botões ou eixos de um controle.

Flexibilidade Total

Essa abstração é vital, pois desvincula a lógica do jogo do hardware específico, permitindo que o mesmo jogo funcione com teclado e mouse, um gamepad ou até mesmo controles de movimento.

Com o Character Movement Component no lugar, nosso personagem tem o potencial de se mover, mas como ele sabe o que fazer? É aqui que entra o sistema de Input, a ponte essencial que conecta as ações físicas do jogador (pressionar uma tecla, mover um joystick) com as ações digitais dentro do jogo. Sem um sistema de input bem configurado, seu personagem seria como um carro sem volante ou pedais: um monte de tecnologia inerte.

Pense no sistema de Input como um tradutor universal de comandos. Ele pega a linguagem do seu teclado (por exemplo, "tecla W pressionada") e a converte para a linguagem do jogo (por exemplo, "ação 'MoverParaFrente' ativada com valor 1.0"). Essa camada de tradução não só facilita o desenvolvimento, mas também oferece flexibilidade para o jogador personalizar seus controles, um recurso cada vez mais valorizado em jogos modernos e um pilar da acessibilidade.

Configurando Input Actions (Ações)

01

Defina o Nome da Ação

Escolha um nome descritivo como "Jump", "Fire" ou "Interact"

02

Associe Teclas ou Botões

Mapeie uma ou mais teclas para ativar a ação (ex: Spacebar, Gamepad Button)

03

Configure Eventos

Defina o que acontece ao pressionar (OnPressed) e soltar (OnReleased)

Dentro do sistema de Input, existem duas categorias principais de mapeamento: as Ações (Actions) e os Eixos (Axes). Vamos começar pelas Ações. Uma Input Action representa um evento discreto, uma ação que tem um início e um fim claros. Pense em ações como "Pular", "Atirar", "Interagir" ou "Abrir Menu". São comandos que geralmente são ativados ao pressionar um botão e podem ter um efeito imediato ou durar enquanto o botão estiver pressionado.

Essas ações são como interruptores de luz. Quando você pressiona o botão, a luz acende (a ação é ativada); quando você solta, a luz apaga (a ação é desativada). No contexto do jogo, isso se traduz em eventos como OnPressed (quando o botão é pressionado) e OnReleased (quando o botão é solto). Essa distinção permite que você crie lógicas diferentes para cada estado, como iniciar um ataque ao pressionar e finalizar a animação ao soltar.

Configurar uma Input Action envolve escolher um nome para a ação (ex: Jump), e então associar uma ou mais teclas ou botões a ela (ex: Spacebar, Gamepad Face Button Bottom). Essa flexibilidade é crucial para permitir que os jogadores personalizem seus controles ou para suportar diferentes layouts de teclado e gamepad. É a maneira mais direta de fazer o personagem reagir a comandos pontuais e decisivos do jogador.

Configurando Axis Mappings (Eixos)

Enquanto as Input Actions lidam com eventos discretos, os Axis Mappings são projetados para movimentos contínuos e graduais. Eles são ideais para controlar a movimentação do personagem (frente/trás, esquerda/direita), a rotação da câmera ou a intensidade de uma ação. Em vez de um simples "ligar/desligar", um Axis Mapping retorna um valor numérico, geralmente entre -1 e 1, que indica a direção e a intensidade do input.

Tecla 'W'

Scale: +1.0 (para frente)

Joystick Analógico

Valores graduais de -1.0 a +1.0

Tecla 'S'

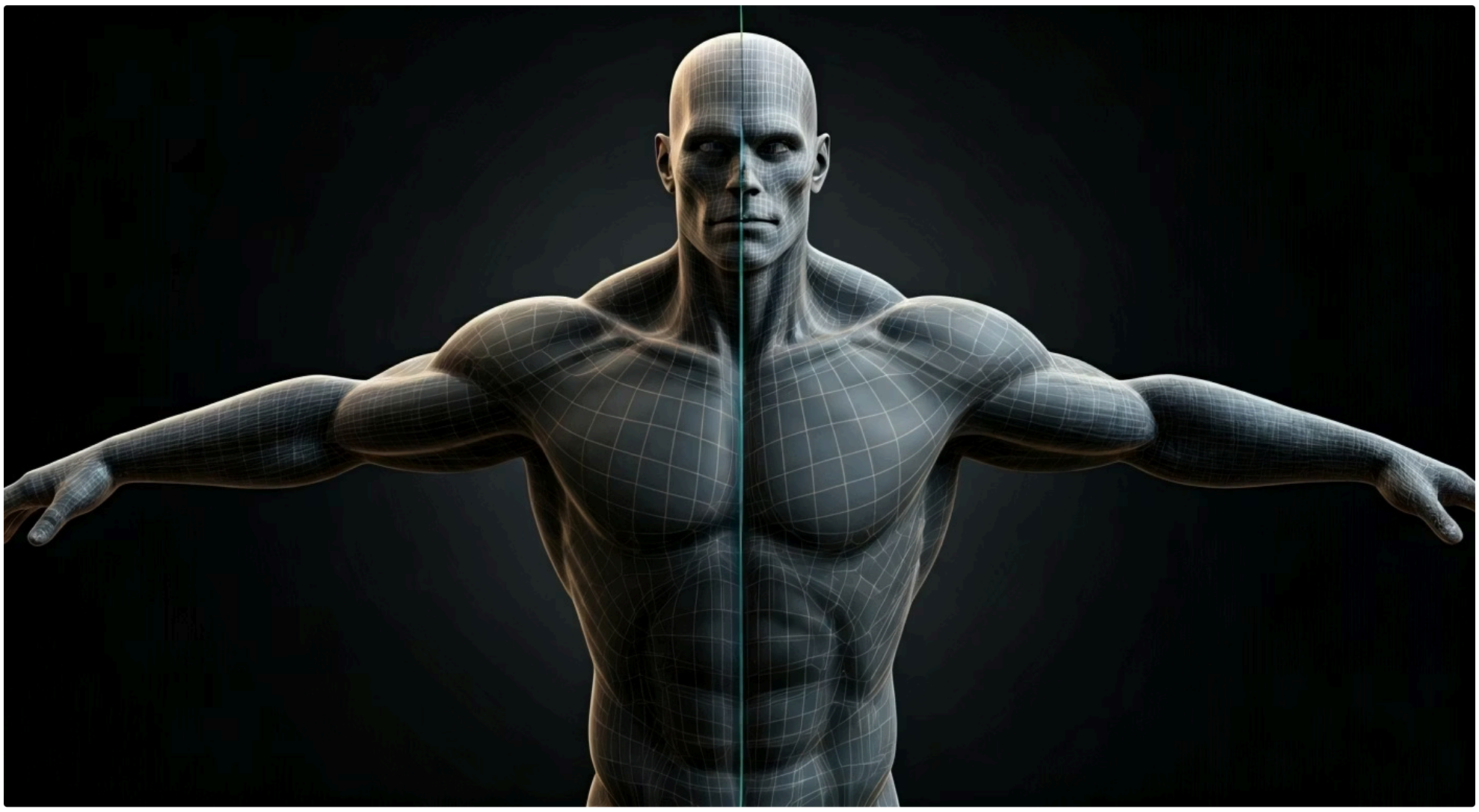
Scale: -1.0 (para trás)

Imagine o volante de um carro ou o joystick de um gamepad. Você não apenas "liga" ou "desliga" o movimento; você o direciona e o modula. Mover o joystick para a frente retorna um valor positivo (ex: 1.0), para trás um valor negativo (ex: -1.0), e para as laterais, valores intermediários. A intensidade com que você move o joystick também afeta o valor, permitindo um controle mais preciso e nuances no movimento do personagem.

Configurar um Axis Mapping envolve nomear o eixo (ex: MoveForward), e então associar teclas ou eixos de gamepad a ele, especificando um "Scale" (escala). Por exemplo, a tecla 'W' pode ser mapeada para MoveForward com Scale 1.0 (para frente), e a tecla 'S' para MoveForward com Scale -1.0 (para trás). Essa abordagem permite que múltiplos inputs contribuam para o mesmo eixo, resultando em uma movimentação fluida e intuitiva.

Conceito	Âmbito/Aplicação	Base/Origem	Exemplo
Input Action	Eventos discretos, ações pontuais	Pressionar/soltar um botão	Pular, Atirar, Interagir, Abrir Menu
Axis Mapping	Movimentos contínuos, graduais, direcionais	Posição de um joystick, teclas direcionais	Mover personagem, Rotacionar câmera, Acelerar

Criando um Personagem em Terceira Pessoa: O Modelo Base



Com os sistemas de movimentação e input em mente, é hora de dar forma ao nosso protagonista. Criar um personagem em terceira pessoa envolve mais do que apenas ter um modelo 3D bonito; é sobre integrar esse modelo com a lógica de jogo para que ele se mova, reaja e interaja de forma convincente. O primeiro passo é o modelo base: o esqueleto, a malha (mesh) e as animações que darão vida ao seu avatar.

1	2	3
Malha 3D (Mesh)	**Esqueleto (Rig)**	**Animações**
Define a forma visual e as texturas do personagem	Estrutura de ossos que permite a animação da malha	Sequências de poses que criam a ilusão de movimento

O modelo 3D do personagem é a sua representação visual no mundo do jogo. Ele é composto por uma malha, que define a forma e a textura, e um esqueleto (rig), que permite que a malha seja animada. As animações, por sua vez, são sequências de poses que dão a ilusão de movimento – andar, correr, pular, atacar. A qualidade desses ativos é fundamental para a imersão, mas a forma como eles são integrados é o que realmente importa.

Pense no modelo base como um ator com seu figurino e um roteiro de movimentos. O ator (o modelo 3D) tem uma aparência específica, e o figurino (as texturas e materiais) aprimora essa aparência. O roteiro (as animações) dita como ele se move e expressa. No motor de jogo, importamos esses ativos e os preparamos para receber os comandos do jogador, garantindo que a transição entre as animações seja suave e que o personagem responda de forma natural aos inputs.

O Blueprint/Prefab do Personagem em Terceira Pessoa

Depois de ter o modelo 3D e as animações, o próximo passo é montar o personagem dentro do motor de jogo. Em motores como Unreal Engine, isso é feito através de um "Blueprint" (um asset visual de programação), e no Unity, através de um "Prefab" (um objeto de jogo pré-configurado e reutilizável). Esses são os contêineres onde todas as peças do nosso personagem são reunidas e configuradas para funcionar em conjunto.

Dentro do Blueprint ou Prefab do personagem, você irá anexar o modelo 3D (a malha e o esqueleto), o Character Movement Component (para gerenciar a movimentação física), e configurar a lógica para processar os Inputs do jogador. É como montar um robô complexo, onde cada peça tem uma função específica, e todas precisam ser conectadas corretamente para que o robô possa executar suas tarefas.

Modelo 3D

Malha + Esqueleto

Movement Component

Física e Colisões

Lógica de Input

Comandos do Jogador

Essa abordagem modular é incrivelmente poderosa. Ela permite que você crie uma "receita" completa para o seu personagem, que pode ser facilmente duplicada ou modificada. Se você precisar de um novo tipo de inimigo com movimentação similar, basta criar uma variação do seu Blueprint/Prefab. Essa reutilização e organização são pilares de um pipeline de produção eficiente, economizando tempo e garantindo consistência em todo o seu projeto.

Lógica de Movimentação: Implementando o Input

Com o Blueprint/Prefab do personagem montado e os Inputs configurados, é hora de fazer a mágica acontecer: conectar os comandos do jogador à movimentação do personagem. Esta é a etapa onde a lógica de programação (seja em código ou visualmente em Blueprints) entra em ação, traduzindo os valores dos Axis Mappings em movimentos reais para o Character Movement Component.

Jogador Move Joystick

Input físico é detectado pelo sistema

Axis Mapping Retorna Valor

MoveForward retorna valor entre -1 e 1

Add Movement Input

Valor é aplicado ao Character Movement Component

Personagem se Move

Movimento físico acontece no mundo do jogo

Para a movimentação, geralmente usamos os valores dos Axis Mappings (que variam de -1 a 1) para determinar a direção e a intensidade do movimento. Por exemplo, quando o jogador move o joystick para frente, o Axis Mapping MoveForward retorna um valor positivo. Esse valor é então passado para uma função do Character Movement Component, como Add Movement Input, que aplica a força necessária para mover o personagem na direção desejada.

Pense nisso como um maestro regendo uma orquestra de ações. O input do jogador é a batuta do maestro, e as notas musicais são os valores dos Axis Mappings. O maestro (sua lógica de jogo) interpreta essas notas e as passa para os músicos (o Character Movement Component), que então executam a melodia (o movimento do personagem). A precisão e a fluidez dessa execução são cruciais para a sensação de controle e a imersão do jogador.

Lógica de Câmera: A Visão do Jogador

A movimentação do personagem é apenas metade da equação; a outra metade é como o jogador vê o mundo através dos olhos (ou sobre os ombros) do personagem. A lógica de câmera é tão crucial quanto a movimentação, pois uma câmera mal implementada pode causar tontura, frustração e quebrar completamente a imersão, mesmo que o personagem se mova perfeitamente.

Spring Arm Component

Atua como uma "vara de pesca" invisível que conecta a câmera ao personagem, permitindo movimento suave e evitando colisões com obstáculos.

Distância Ideal

Mantém a câmera a uma distância estratégica do personagem, ajustando-se dinamicamente aos movimentos e garantindo visibilidade.

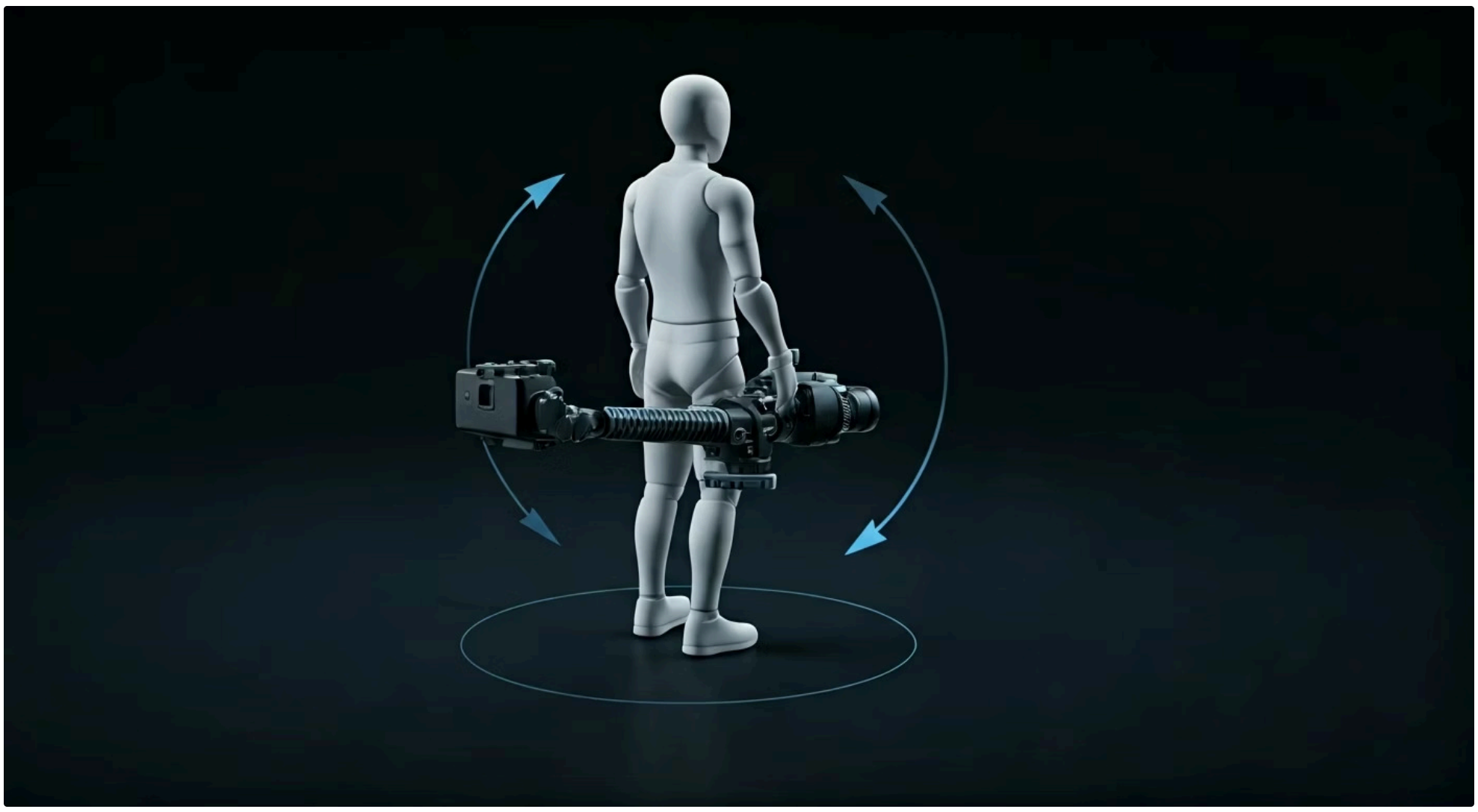
Foco Constante

Garante que a ação principal esteja sempre em foco, sem obstruções, proporcionando a melhor perspectiva possível.

Em jogos de terceira pessoa, a câmera geralmente segue o personagem a uma certa distância, permitindo que o jogador veja o avatar e o ambiente ao redor. Para conseguir um movimento suave e responsivo, motores de jogo oferecem componentes como o Spring Arm (Unreal Engine) ou sistemas como o Cinemachine (Unity). O Spring Arm, por exemplo, é um componente que atua como uma "vara de pesca" invisível, conectando a câmera ao personagem e permitindo que ela se mova suavemente, evitando colisões com obstáculos e mantendo uma distância ideal.

Imagine um cinegrafista experiente seguindo o ator principal em uma cena de ação. Ele não está colado ao ator, mas mantém uma distância estratégica, ajustando-se aos movimentos e garantindo que a ação principal esteja sempre em foco, sem obstruções. Essa é a essência da lógica de câmera: proporcionar ao jogador a melhor perspectiva possível, de forma orgânica e sem distrações, para que ele possa se concentrar na jogabilidade.

Controle de Câmera: Rotação e Perspectiva



Além de seguir o personagem, a câmera em jogos de terceira pessoa precisa permitir que o jogador olhe ao redor. Isso é geralmente controlado pelo mouse (em PC) ou pelo segundo joystick (em gamepads), que manipulam a rotação da câmera em torno do personagem. Essa liberdade de perspectiva é fundamental para a exploração, a mira e a consciência situacional do jogador.

Yaw (Rotação Horizontal)

Controlado pelo movimento horizontal do mouse ou joystick. Permite girar a câmera ao redor do personagem.

Pitch (Rotação Vertical)

Controlado pelo movimento vertical do mouse ou joystick. Permite olhar para cima e para baixo.

A rotação da câmera é mapeada para Axis Mappings, assim como a movimentação do personagem. Mover o mouse para a esquerda/direita ou o joystick horizontalmente afeta o Yaw (rotação horizontal) da câmera, enquanto mover para cima/baixo ou o joystick verticalmente afeta o Pitch (rotação vertical). É importante que esses movimentos sejam suaves e que a sensibilidade possa ser ajustada pelo jogador, para se adequar às suas preferências.

Dica Profissional

Sempre ofereça opções para ajustar a sensibilidade da câmera e permitir inversão dos eixos. Isso melhora significativamente a acessibilidade e a experiência do jogador.

Essa capacidade de controlar a câmera é como ter o pescoço do jogador dentro do jogo. Ele pode virar a cabeça para inspecionar um detalhe no cenário, mirar em um inimigo distante ou simplesmente apreciar a vista. Uma boa implementação do controle de câmera não apenas oferece liberdade, mas também contribui para a sensação de agência e imersão, fazendo com que o jogador se sinta verdadeiramente conectado ao mundo virtual.

Desafios Comuns e Otimização

Personagem "Escorregadio"

Ajuste a aceleração e desaceleração no Character Movement Component para melhorar a resposta.

Câmera Atravessa Paredes

Configure o Spring Arm para detectar colisões e empurrar a câmera para fora de obstáculos.

Inputs Não Registram

Verifique os mapeamentos de input e certifique-se de que os eventos estão conectados corretamente.

Pulo Inconsistente

Ajuste a força do pulo e a gravidade para criar uma sensação mais previsível e satisfatória.

Mesmo com os componentes e sistemas bem definidos, o desenvolvimento da movimentação e do input raramente é um processo sem percalços. É comum encontrar desafios que exigem depuração e otimização. Um personagem pode parecer "escorregadio", a câmera pode atravessar paredes ou os inputs podem não registrar corretamente. Identificar e resolver esses problemas é uma parte crucial do polimento do jogo.

Um dos problemas mais frequentes é o ajuste fino dos parâmetros do Character Movement Component. Se a aceleração for muito baixa e a velocidade máxima muito alta, o personagem pode parecer que patina. Se a força do pulo for inconsistente, a navegação em plataformas se torna frustrante. Outro desafio é lidar com colisões inesperadas da câmera, que podem ser resolvidas ajustando o Spring Arm ou implementando lógica para empurrar a câmera para fora de obstáculos.

A solução para esses desafios reside na iteração e nos testes contínuos. Use as ferramentas de depuração do motor de jogo para visualizar vetores de movimento, colisões e eventos de input. Peça a outras pessoas para testarem seu jogo e coletar feedback sobre a sensação da movimentação e do controle. Pense nisso como refinar a calibração de um instrumento musical: são os pequenos ajustes que transformam um som bom em uma melodia perfeita, garantindo que a experiência do jogador seja fluida e agradável.

Tendências e Boas Práticas em Movimentação e Input

O campo do desenvolvimento de jogos está em constante evolução, e a movimentação e o input não são exceção. As game engines modernas, como Unity e Unreal Engine, estão sempre introduzindo novas ferramentas e abordagens para tornar esses sistemas mais robustos, flexíveis e acessíveis. Manter-se atualizado com essas tendências e adotar boas práticas é fundamental para criar jogos competitivos e de alta qualidade.

Sistemas de Input Avançados

Unity Input System e Unreal Enhanced Input oferecem maior flexibilidade para remapear controles e suportar múltiplos dispositivos.

Feedback Audiovisual

Implemente sons de passos, animações de impacto e efeitos visuais para enriquecer cada ação do jogador.

Acessibilidade em Primeiro Lugar

Permita personalização completa de controles, ajuste de sensibilidade e remapeamento de funções para todos os jogadores.

Uma tendência notável é a ascensão de sistemas de input mais avançados, como o Input System no Unity e o Enhanced Input System no Unreal Engine. Esses sistemas oferecem maior flexibilidade para remapear controles, suportar múltiplos dispositivos e lidar com inputs complexos de forma mais elegante. Outra boa prática é a implementação de feedback visual e sonoro para cada ação, como um som de passo ao andar ou uma animação de impacto ao atacar, o que enriquece a experiência do jogador.

Além disso, a acessibilidade se tornou uma prioridade. Isso significa permitir que os jogadores personalizem completamente seus controles, ajustem a sensibilidade, invertam eixos e até mesmo remapeiem funções para diferentes botões. Pense na evolução dos controles de um carro: de alavancas e pedais básicos a sistemas eletrônicos sofisticados que se adaptam ao motorista. Adotar essas práticas não apenas melhora a qualidade do seu jogo, mas também o torna mais inclusivo e agradável para um público mais amplo.

Integrando Tudo: Do Conceito à Jogabilidade

Chegamos ao ponto crucial onde todas as peças se encaixam. Vimos o Character Movement Component como o motor do nosso personagem, o sistema de Input como a ponte entre o jogador e o jogo, a criação do modelo base e a montagem do Blueprint/Prefab, e a lógica de câmera que define a perspectiva do jogador. Agora, o desafio é garantir que todos esses elementos trabalhem em harmonia, transformando conceitos técnicos em uma experiência de jogabilidade coesa e envolvente.

A integração não é apenas sobre conectar fios; é sobre a sinergia entre cada componente. A velocidade do personagem deve se alinhar com suas animações, a sensibilidade da câmera deve complementar a agilidade do movimento, e os inputs devem ser responsivos sem serem excessivamente sensíveis. É um processo de ajuste fino contínuo, onde cada pequena alteração pode ter um impacto significativo na sensação geral do jogo.

Pense nisso como a orquestra completa tocando uma sinfonia. Cada músico (componente) domina seu instrumento, mas é a coordenação e a direção do maestro (o desenvolvedor) que transformam notas individuais em uma melodia harmoniosa. Testes de jogabilidade regulares, com foco específico na movimentação e no controle, são essenciais para identificar gargalos e refinar a experiência, garantindo que a visão do seu jogo se traduza em uma realidade divertida e imersiva para o jogador.

Consolidação e Próximos Passos

Chegamos ao fim de nossa exploração sobre movimentação de personagem e input. Vimos como o Character Movement Component atua como o cérebro da locomoção, como os sistemas de Input (Actions e Axes) traduzem os comandos do jogador, e como a lógica de câmera molda a perspectiva. Entender e dominar esses conceitos é fundamental para qualquer desenvolvedor de jogos, pois eles formam a base da interação do jogador com o mundo virtual.

Em prática

Para solidificar seu aprendizado, experimente criar um novo projeto em sua game engine preferida. Configure um personagem básico em terceira pessoa, implemente os inputs para movimentação e pulo, e ajuste a câmera para seguir o personagem de forma suave. Brinque com os parâmetros do Character Movement Component e observe como pequenas mudanças afetam a sensação de controle.

Autoavaliação

1. Qual a principal função do Character Movement Component em um motor de jogo?
   - a) Gerenciar apenas as animações do personagem.
   - b) Controlar a interface de usuário (UI) do jogo.
   - c) Orquestrar a movimentação física, colisões e interações do personagem com o ambiente.
   - d) Definir apenas a aparência visual do personagem.
2. A diferença fundamental entre Input Actions e Axis Mappings é que:
   - a) Input Actions são para eventos contínuos, e Axis Mappings para eventos discretos.
   - b) Input Actions retornam valores entre -1 e 1, enquanto Axis Mappings são apenas "ligar/desligar".
   - c) Input Actions lidam com eventos discretos (ex: pular), e Axis Mappings com movimentos graduais e contínuos (ex: mover).
   - d) Não há diferença, são termos sinônimos.
3. Qual componente é comumente utilizado em game engines para garantir que a câmera siga o personagem de forma suave e evite colisões com obstáculos?
   - a) Collision Box
   - b) Static Mesh Component
   - c) Spring Arm (ou sistemas similares como Cinemachine)
   - d) Audio Component
4. Ao configurar a movimentação de um personagem, qual dos seguintes parâmetros é mais provável de ser ajustado no Character Movement Component para alterar a "sensação" de peso e agilidade?
   - a) Cor do material do personagem.
   - b) Velocidade máxima de caminhada e aceleração.
   - c) Número de polígonos do modelo 3D.
   - d) Tipo de fonte do texto na tela.
5. Explique a importância da personalização dos controles (remapeamento de teclas/botões) para a acessibilidade e a experiência do jogador em jogos modernos.

Gabarito

1 **c) Orquestrar a movimentação física, colisões e interações do personagem com o ambiente.**

2 **c) Input Actions lidam com eventos discretos (ex: pular), e Axis Mappings com movimentos graduais e contínuos (ex: mover).**

3 **c) Spring Arm (ou sistemas similares como Cinemachine)**

4 **b) Velocidade máxima de caminhada e aceleração.**

Recursos e Próxima Aula

Próxima Aula

Aula 24 – Físicas, Colisões e Triggers

Aprofundaremos ainda mais a interação do personagem com o mundo, explorando como objetos reagem à força, como detectar contatos e como criar áreas de ativação para eventos específicos.

Recursos Adicionais

- Documentação oficial do Unreal Engine sobre Character Movement Component: Para detalhes técnicos aprofundados.
- Documentação oficial do Unity sobre Character Controller e Input System: Para explorar as ferramentas equivalentes.
- Tutoriais em vídeo sobre criação de personagens em terceira pessoa: Para ver o processo na prática.

NOTA IMPORTANTE

As informações técnicas desta aula estão atualizadas até 2025. Consulte sempre as documentações oficiais das game engines para verificar alterações e novas funcionalidades.