ROBLOX R6

Procedural IK System

BULLET STUDIO· Commercial Edition

Versão
1.0.0

Data
Abril 2026

Compatibilidade
Roblox R6 · Client-Side

Tipo de Licença
Comercial · Single-Game

1. Visão Geral do Produto

O R6 Procedural IK System é um plugin de animação procedural completo para personagens R6 no Roblox. Diferente das animações em keyframe tradicionais, o sistema calcula a posição dos pés em tempo real usando Cinemática Inversa (IK), fazendo o personagem reagir dinamicamente ao terreno — escadas, rampas, superfícies irregulares — sem nenhuma animação pré-criada.

O produto é comercializado como asset pronto para uso (plug-and-play) para developers Roblox que desejam elevar o nível de qualidade visual dos seus jogos sem desenvolver um sistema de IK do zero.

1.1 Proposta de Valor

✅	**Pronto para uso imediato**	Plug-and-play: arraste para o jogo e funciona
⚡	**Alta performance**	Otimizado para múltiplos rigs simultâneos
🎮	**Controles de corrida integrados**	Run procedural com SHIFT — lerp suave e configurável
🔧	**Totalmente configurável**	Tabelas WALK / RUN com 5 parâmetros cada
🧼	**Código limpo e documentado**	Arquitetura modular com 3 módulos independentes
🔁	**Cleanup automático**	Sem memory leaks: desconecta ao morrer ou respawn

1.2 Público-alvo

- Developers intermediários e avançados de jogos Roblox
- Estudantes de game design buscando aprender sistemas de IK
- Estúdios que produzem jogos de aventura, RPG ou simulação
- Criadores de experiências com personagens humanoides em terrenos complexos

2. Arquitetura Técnica

O sistema é dividido em três módulos com responsabilidades bem definidas, comunicando-se por uma interface simples. O LocalScript atua como orquestrador, criando a cadeia de IK e acionando o loop de animação a cada frame.

2.1 Diagrama de Módulos

IKController (LocalScript — Orquestrador)

Detecta entrada (SHIFT), calcula runAlpha, ajusta WalkSpeed e chama animator:Animate(dt, runAlpha) no Heartbeat.

├── **ProceduralAnimator (ModuleScript)**

│ Gerencia o ciclo de passos, bob do corpo, inclinação forward e alvo dos pés via lerp entre WALK e RUN.

└── **CCDIKController (ModuleScript)**

Resolve a cadeia de Motor6D usando o algoritmo CCD (Cyclic Coordinate Descent) com limite de iterações.

2.2 Cadeia de IK — R6

O R6 original usa dois Motor6Ds (Left Hip / Right Hip) diretamente do Torso para as pernas. Para possibilitar IK com dois segmentos (coxa + perna), o sistema interpõe uma FakePart invisível entre o Torso e cada perna, criando uma cadeia de 2 motores por perna.

Segmento	Motor6D	Descrição
`Torso → FakeUpperLeg`	Fake Hip	Rotação do quadril (BallSocketConstraint)
`FakeUpperLeg → Leg`	Fake Leg	Flexão do joelho (HingeConstraint)
`Leg → EndEffector`	—	Ponto de contato com o chão (Attachment)

2.3 Constraints Físicas

Constraints são usadas para restringir o movimento a ângulos anatomicamente plausíveis, evitando que o IK produza poses impossíveis:

- HingeConstraint no joelho: ângulos entre −145° e −15° (flexão somente)
- BallSocketConstraint no quadril: UpperAngle = 5°, TwistLimitsEnabled = true
- Os Motors originais Left Hip / Right Hip ficam Enabled = false

3. Features Detalhadas

3.1 Procedural Walk

A animação de caminhada é inteiramente calculada em tempo real. Não existe nenhum AnimationTrack ativo. A cada frame o sistema:

1. Avança o ciclo de passo em radianos: CurrentCycle = (CurrentCycle + stepSpeed × dt) % 2π
2. Calcula o alvo do pé (footTarget) com base no ciclo, strideZ e stepHeight
3. Executa um raycast do quadril até o chão para encontrar a superfície real
4. Resolve a cadeia IK via CCDIKController até o alvo (máximo 8 iterações)
5. Aplica bob vertical e inclinação no RootJoint

3.2 Procedural Run (SHIFT Hold)

O sistema de corrida interpola suavemente entre dois conjuntos de parâmetros (WALK e RUN) usando um valor de alpha de 0 a 1 calculado a cada frame:

runAlpha lerpa 0→1 enquanto SHIFT pressionado + personagem em movimento
WalkSpeed = 16 + (24 − 16) × runAlpha *[16 walk → 24 run]*
Todos os parâmetros de animação são lerpados via lerpConfig(WALK, RUN, runAlpha)

Parâmetros interpolados walk → run

Parâmetro	Walk	Run	Efeito visual
`stepSpeed`	2.5 ciclos/s	5.2 ciclos/s	Frequência do passo — mais rápido ao correr
`strideZ`	0.30 studs	0.62 studs	Comprimento da passada
`stepHeight`	0.15 studs	0.32 studs	Altura do arco do pé
`bodyBobY`	0.04 studs	0.10 studs	Amplitude do bob vertical do tronco
`leanX`	0°	8° (radianos)	Inclinação forward do RootJoint

3.3 Detecção de Superfície (Raycast)

A cada frame, um raycast é disparado da posição do quadril de cada perna em direção ao chão. O personagem inteiro é excluído do FilterDescendantsInstances para evitar auto-colisão. O footTarget é posicionado no ponto de hit, fazendo o pé pousar sobre o terreno real.

3.4 Sistema de Som

O sound Running nativo do HumanoidRootPart tem seu volume zerado para evitar conflito com o som procedural. O sistema expõe um slot ObjectValue chamado FootStepSound para que o developer conecte um Sound customizado de passos via animator:ConnectFootStepSound().

3.5 Cleanup Automático

A conexão de Heartbeat é armazenada e desconectada automaticamente em dois eventos, garantindo zero memory leaks mesmo em jogos com respawn frequente:

- Humanoid.Died → animationConnection:Disconnect()
- HumanoidRootPart.AncestryChanged (parent == nil) → animationConnection:Disconnect()

4. Guia de Configuração

Todos os parâmetros de animação estão centralizados nas tabelas WALK e RUN no topo do ProceduralAnimator. Nenhuma linha de código precisa ser alterada fora dessas tabelas para customizar o feel da animação.

4.1 Velocidades (LocalScript)

Constante	Padrão	Descrição
`WALK_SPEED`	16	WalkSpeed ao caminhar (padrão Roblox)
`RUN_SPEED`	24	WalkSpeed ao correr com SHIFT
`LERP_SPEED`	8	Velocidade da transição walk↔run em s^{-1} (~125ms)

4.2 Parâmetros de Animação (ProceduralAnimator)

Parâmetro	Walk	Run	Dica de tuning
`stepSpeed`	2.5	5.2	Sobe se parecer deslizando; desce se apressado
`strideZ`	0.30	0.62	Ajuste ±0.05 se pés flutuarem ou enterrarem
`stepHeight`	0.15	0.32	Valores >0.50 ficam exagerados
`bodyBobY`	0.04	0.10	0.04–0.06 walk, 0.08–0.14 run soa natural
`leanX`	0 rad	0.14 rad	5°–10° natural; acima de 15° fica caricato

4.3 Limites do CCDIK

O número máximo de iterações do solver CCD é configurado diretamente no CCDIKController. O padrão recomendado é:

- 8 iterações: qualidade máxima, adequado para poucos rigs (<10 simultâneos)
- 4–6 iterações: melhor custo-benefício para lobbies com muitos jogadores
- 2–3 iterações: modo de emergência para servidores com >30 rigs simultâneos

5. Instalação

5.1 Requisitos

- Roblox Studio (versão atual)
- Personagem R6 — NÃO compatível com R15 nesta versão
- FilteringEnabled = true (padrão em todos os jogos novos)
- O LocalScript deve ser filho de StarterCharacterScripts ou StarterPlayerScripts

5.2 Passo a Passo

6. Importe o arquivo do rbxm. Ele contém: IKController (LocalScript), ProceduralAnimator (ModuleScript) e CCDIKController (ModuleScript).
7. Mova IKController para StarterCharacterScripts.
8. ProceduralAnimator e CCDIKController devem ser filhos do IKController.
9. (Opcional) Crie um ObjectValue chamado FootStepSound dentro do IKController e aponte-o para o Sound de passos desejado.
10. Pressione Play. Os passos procedurais devem funcionar imediatamente.
11. Segure SHIFT para ativar a corrida procedural.

5.3 Verificação

Para confirmar que o sistema está ativo, abra o Output e verifique que não há erros. No jogo, observe:

- Pés do personagem pousando na superfície do terreno ao caminhar
- Bob vertical sutil no tronco durante o movimento
- Transição suave de animação ao pressionar/soltar SHIFT
- Nenhum som de Running nativo (substituído pelo FootStepSound customizado)

6. Referência de API

ProceduralAnimator

Método / Propriedade	Assinatura	Descrição
`.new()`	ProceduralAnimator.new(rootPart, legs, rootMotor, raycastParams)	Cria uma nova instância do animator
`:Animate()`	:Animate(dt: number, runAlpha: number?)	Processa um frame de animação. runAlpha é opcional (default 0)
`:ConnectFootStepSound()`	:ConnectFootStepSound(sound: Sound)	Conecta um Sound para tocar a cada passo

CCDIKController

Método / Propriedade	Assinatura	Descrição
`.new()`	CCDIKController.new(motors: {Motor6D})	Cria controlador para a cadeia de motores fornecida
`.UseLastMotor`	boolean (default: true)	Se true, aplica a solução no motor mais distal da cadeia
`:GetConstraints()`	:GetConstraints()	Lê HingeConstraints e BallSocketConstraints na cadeia
`:GetConstraintsFromMotor()`	:GetConstraintsFromMotor(motor, name)	Vincula um BallSocketConstraint pelo nome ao motor-raiz
`:Solve(target)`	:Solve(target: CFrame)	Executa o solver CCD até o alvo com o budget de iterações

7. Licença Comercial

7.1 Termos de Uso

- O comprador da licença pode usar o sistema Permanentimente.
- A licença não permite redistribuição do código-fonte ou revenda como asset independente.
- Modificações são permitidas para uso pessoal dentro do jogo licenciado.
- A Source License é a única modalidade que permite redistribuição ou uso em frameworks vendidos.
- Roblox Corporation não é parceira deste produto. O sistema opera dentro das APIs públicas da plataforma.

8. Roadmap

v1.0 (Atual)

- Procedural Walk via CCDIK com 2 segmentos por perna
- Run procedural com SHIFT hold e lerp suave
- Constraints de joelho (HingeConstraint) e quadril (BallSocketConstraint)
- Cleanup automático e zero memory leaks
- Som de passos via ObjectValue configurável

v1.1 (Planejado — Q3 2026)

- Parâmetro de inclinação lateral do tronco em curvas (side lean)
- Integração com Humanoid.MoveDirection para antecipar os passos
- Modo debug visual (helper parts coloridas mostrando targets IK)

v2.0 (Planejado — Q1 2027)

- Compatibilidade com R15 (cadeia de 3 segmentos, 6 motores por perna)
- Sistema de estados explícitos: Idle / Walk / Run / Crouch / Sprint
- API de callback por evento de passo (OnStep) para partículas e som avançado
- Modo multi-character: pool de raycasts para otimizar servidores com >20 rigs

9. FAQ e Troubleshooting

Perguntas Frequentes

P: Funciona com personagem R15?
R: Não nesta versão. R15 possui cadeia de 3 segmentos e 6 motores por perna, o que exige uma versão dedicada planejada para v2.0.

P: O jogo usa rig customizado baseado em R6. Funciona?
R: Sim, desde que o rig mantenha os nomes padrão: Torso, Left Leg, Right Leg, HumanoidRootPart, Left Hip e Right Hip.

P: Os pés ficam tremendo ou em poses estranhas.
R: Verifique se o CCDIKController está com UseLastMotor = true e se os ângulos dos HingeConstraints (LowerAngle = -145, UpperAngle = -15) estão corretos. Aumentar o MAX_ITER para 10–12 também pode resolver.

P: A animação não inicia — sem erros no Output.
R: Confirme que o IKController está em StarterCharacterScripts. Em StarterPlayerScripts o personagem pode ainda não existir quando o script inicia.

P: SHIFT não ativa a corrida.
R: Certifique que nenhum outro LocalScript está capturando o evento de SHIFT antes deste. UserInputService:IsKeyDown retorna false se o TextBox estiver focado — isso é comportamento padrão do Roblox.