Atividade 3 — Mundo procedural: terreno, chunks, cavernas e itens

Visão geral

Nesta atividade, você construirá um mundo gerado por código: um terreno contínuo feito de pedaços que nascem e somem conforme o jogador anda, com pelo menos uma caverna, itens coletáveis que não voltam depois de coletados, um inventário e um sistema de salvar e carregar. O objetivo é exercitar os conceitos das aulas 1 e 2 de geração procedural, todos juntos, em um projeto pequeno e funcional.

Não há objetivo de jogo, progressão nem inimigos. Isso fica para o trabalho final.

Equipe: de 1 a 3 pessoas.

Duração: 2 semanas, sem encontro presencial. Dúvidas pelo Discord. Há um checkpoint obrigatório ao fim da primeira semana (ver seção *Checkpoint*).

Pré-requisito: as aulas 1 e 2 de geração procedural e a tarefa do quad com ruído. O motor de jogos e a linguagem de programação permanecem de livre escolha; Unity e Godot são os esperados. Se o projeto for continuação do airsoft, o HUD e o jogador da Atividade 2 podem ser reaproveitados.

Ao concluir a atividade, você deverá ser capaz de:

- gerar um heightmap por fBm e transformá-lo em uma malha própria;
- dividir o mundo em chunks e mantê-los consistentes entre si;
- deslocar a origem do mundo sem que o jogador perceba;
- gerar uma caverna por autômato celular;
- distribuir itens por um mapa de densidade e impedir que reapareçam;
- separar o modelo de dados do inventário da sua interface;
- salvar e restaurar o estado do mundo a partir de uma semente.

Arquitetura esperada

O projeto deve ter quatro partes com responsabilidades separadas:

1. **Gerador de ruído.** Recebe uma coordenada global e os parâmetros (semente ou offset, escala, oitavas, persistência, lacunaridade) e devolve um número. Não sabe o que é chunk nem malha.
2. **Gerador de chunk.** Recebe a coordenada de um chunk e produz o heightmap, a malha e a lista de itens daquele chunk, amostrando o ruído em coordenada global. Não sabe quantos chunks existem nem onde o jogador está.
3. **Gerenciador de chunks.** Sabe onde o jogador está, mantém a grade 3x3 ao redor dele, cria e descarta (ou reaproveita de um pool) chunks, e dispara o floating origin.
4. **Estado do mundo.** Guarda a semente, o conjunto de chaves de itens coletados, o inventário e a posição do jogador. É a única parte que vai para o arquivo de save.

A interface (HUD, tela de inventário) apenas apresenta o estado atual; ela não deve ser a fonte dos dados.

Desafio

Crie uma cena em que o jogador anda por um terreno gerado, entra em uma caverna, coleta itens e vê o inventário. Ao sair e voltar de uma região, o terreno é o mesmo e os itens coletados não estão mais lá. Ao fechar o jogo e carregar o save, o mundo é o mesmo.

Especificações obrigatórias

1. Terreno por ruído

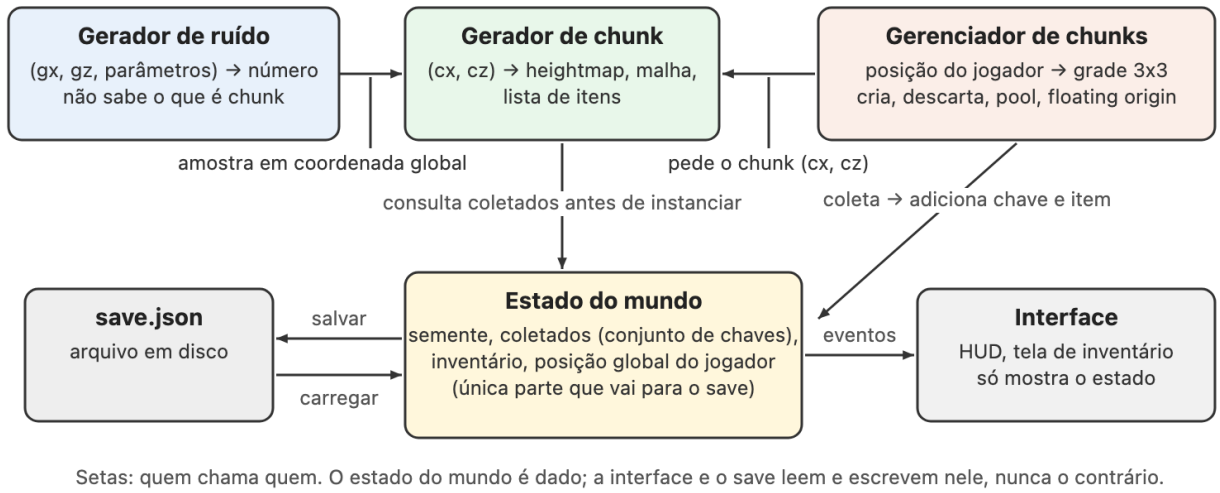

Figura 1: As quatro partes e quem chama quem. O estado do mundo é o único que vai para o save.

Propriedade	Regra
Ruído	Perlin (ou Simplex) somado em pelo menos 3 oitavas (fBm).
Curva de altura	Uma redistribuição aplicada ao valor do ruído antes de usá-lo como altura: potência, `AnimationCurve`, `Curve` ou equivalente.
Faixas de altura	Pelo menos 4: água, areia, campo, rocha. Cada faixa decide cor ou material e se é caminhável.
Parâmetros expostos	Escala, oitavas, persistência, lacunaridade, semente (ou offset, no Unity) e altura máxima, editáveis no Inspector ou em arquivo de configuração.

Mudar qualquer parâmetro e regenerar deve produzir um terreno diferente. Mudar nenhum e regenerar deve produzir o mesmo terreno.

2. Mundo em chunks

Propriedade	Regra
Malha	Cada chunk é uma malha própria (`Mesh` no Unity, `ArrayMesh` ou `SurfaceTool` no Godot). Não use o `Terrain` do Unity.
Tamanho	Fixo e configurável, por exemplo 64×64 vértices.
Grade	Sempre 9 chunks carregados: o chunk do jogador e os 8 vizinhos.
Troca	Quando o jogador entra em outro chunk, os chunks que saíram da grade são descartados ou devolvidos a um pool, e os que faltam são criados.

Propriedade	Regra
Costura	O ruído é amostrado em coordenada global (`chunkX * tamanho + x`), nunca em coordenada local. Não pode haver degrau nem emenda visível entre chunks.

Um chunk que sai da grade e volta deve ser idêntico ao que era.

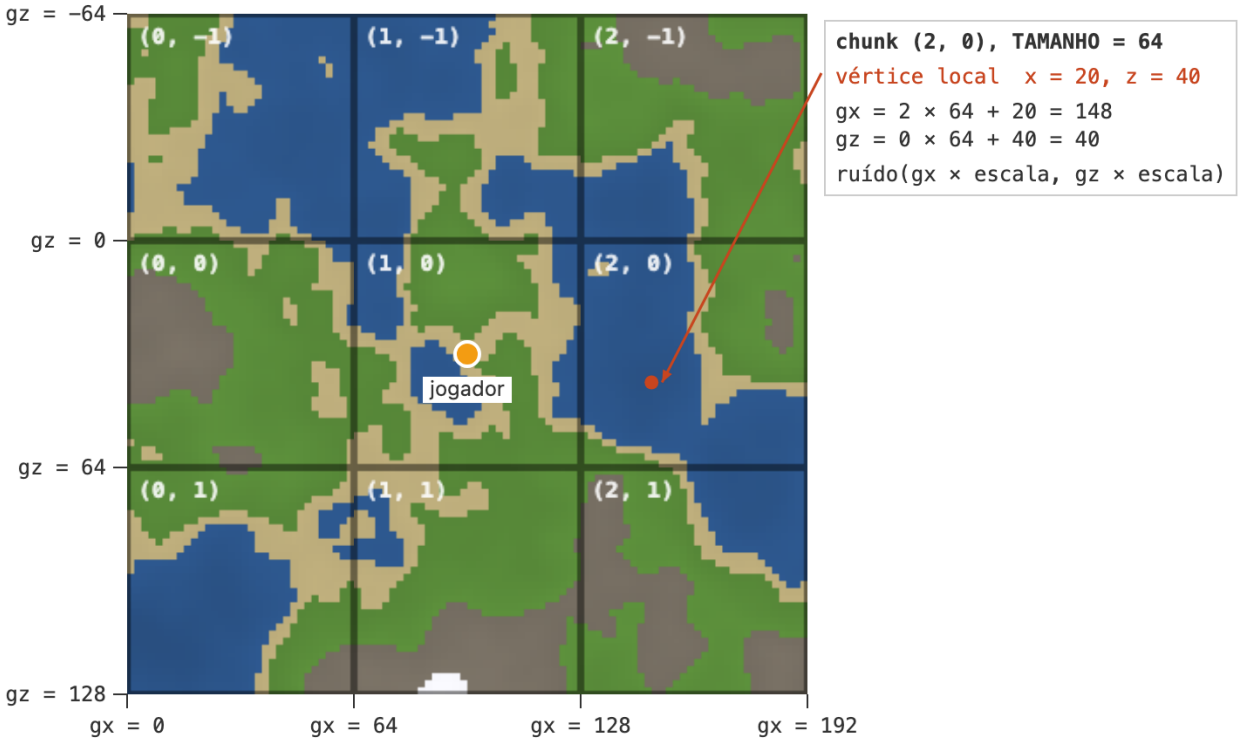

Figura 2: Grade 3x3 ao redor do jogador e a conta da coordenada global de um vértice: $gx = cx \times \text{TAMANHO} + x$.

3. Floating origin

Quando a distância do jogador até a origem passar de um limite configurável (por exemplo, 1000 unidades), desloque todo o mundo, incluindo os chunks, os itens e o próprio jogador, de modo que o jogador volte para perto da origem.

- O deslocamento deve ser invisível para o jogador: sem tranco, sem teleporte perceptível.
- Os chunks continuam sendo gerados a partir da coordenada global correta depois do deslocamento. Guarde o deslocamento acumulado.
- Documente no README quando o deslocamento é disparado e o que é deslocado.

Para verificar: ande até 5000 unidades da origem sem floating origin e observe o tremor da câmera e dos objetos. Com floating origin, o tremor não aparece.

4. Caverna por autômato celular

Pelo menos uma região do mundo deve conter uma caverna gerada por autômato celular:

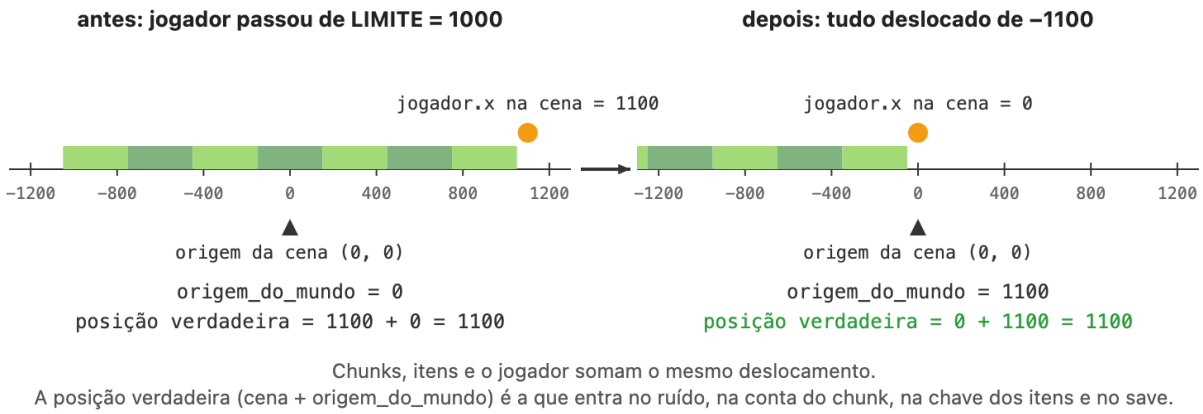

Figura 3: Antes e depois do deslocamento. Na cena o jogador volta para perto de zero; a posição verdadeira continua a mesma porque o deslocamento fica guardado em origem_do_mundo.

1. preencha um grid com paredes e vazios aleatoriamente, usando a semente e uma porcentagem inicial de paredes;
2. aplique a regra de vizinhança de Moore por pelo menos 4 iterações: conte as 8 células ao redor; com mais de 4 paredes a célula vira parede, com menos de 4 vira vazio, com exatamente 4 fica como está;
3. use flood fill para identificar as regiões vazias e remova as menores que um limite;
4. garanta que o jogador consiga entrar na caverna a partir da superfície.

A caverna pode ser 2D (um piso e paredes extrudadas) ou 3D. Alternativa aceita: caverna por ruído 3D com limiar, desde que o README explique a escolha e os parâmetros.

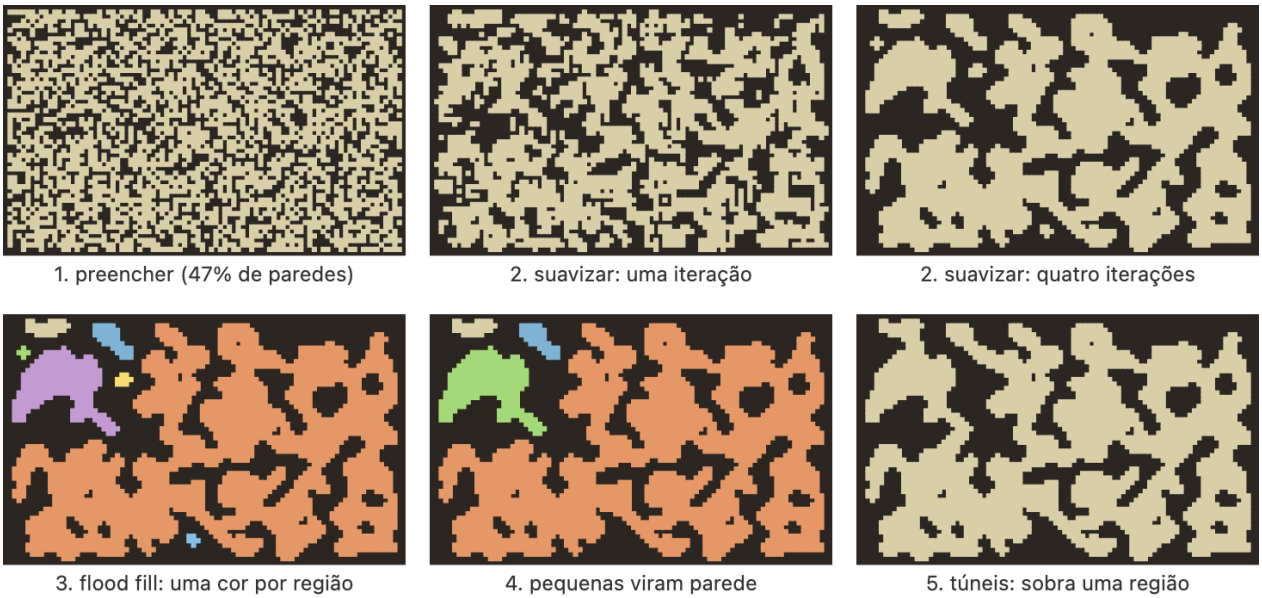

Figura 4: Os cinco passos em uma grade de 90 × 56 células. Nos passos 3 a 5, cada cor é uma região vazia encontrada pelo flood fill.

5. Itens coletáveis com chave

Propriedade	Regra
Distribuição	Ruído como mapa de densidade: o item nasce onde o ruído passa de um limiar. Use um ruído independente do terreno (outra semente ou outro offset).
Posição	Só em faixas caminháveis, pousado no chão (raycast ou altura do heightmap).
Chave	Cada item tem uma chave determinística derivada de (semente, coordenada do chunk, índice do item dentro do chunk). A mesma chave nasce sempre no mesmo lugar.
Coleta	Ao coletar, a chave entra em um conjunto de coletados e o item vai para o inventário.
Recriação	Ao gerar um chunk, itens cuja chave está no conjunto de coletados não são instanciados.

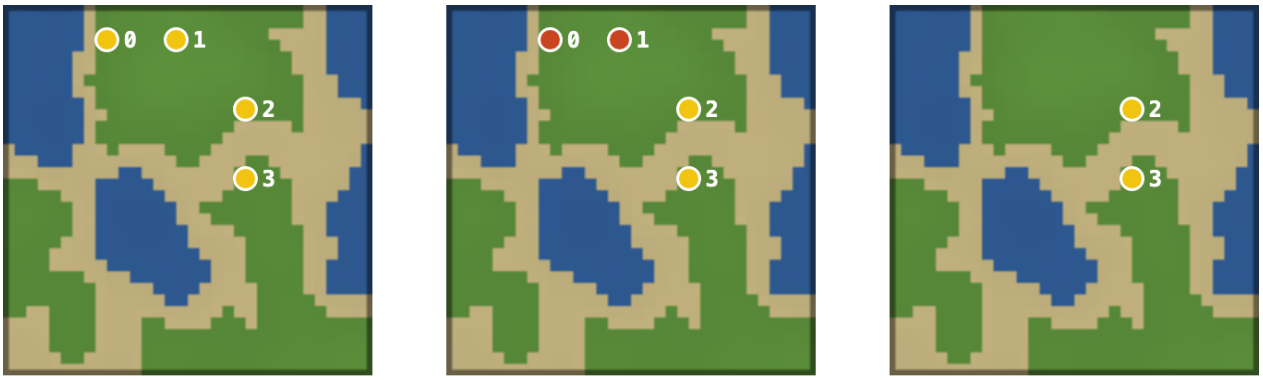

Figura 5: O mesmo chunk gerado, depois da coleta de dois itens, e gerado de novo. O índice do item dentro do chunk avança mesmo para os já coletados, senão as chaves dos outros mudam.

6. Inventário

Propriedade	Regra
Modelo	Uma classe de dados com lista de slots; cada slot tem id do item e quantidade. Capacidade máxima e empilhamento por tipo.
Interface	Uma tela ou painel que mostra os slots. Atualizado por eventos disparados pelo modelo, não por busca de objetos a cada frame.
Integração	Se o projeto for continuação do airsoft, o HUD da Atividade 2 deve mostrar o inventário ou ao menos a contagem de itens.

Adicionar um item além da capacidade deve falhar com retorno ao jogador, sem perder o item do chão.

7. Salvar e carregar

Salve em um arquivo JSON:

- a semente (ou os offsets) do mundo;
- o conjunto de chaves de itens coletados;
- o inventário;
- a posição do jogador em coordenada global (considerando o deslocamento acumulado do floating origin).

Carregar deve reconstruir o mesmo mundo, com o jogador no mesmo lugar e com os itens já coletados ausentes. Uma tecla ou botão para salvar e outro para carregar são suficientes.

Estados que devem ser tratados

Estado	Comportamento esperado
Jogador cruza a borda de um chunk	A grade se atualiza; nada some debaixo do jogador.
Chunk sai da grade e volta	O terreno é idêntico; itens não coletados voltam; coletados não voltam.
Item coletado e chunk recriado	O item não reaparece.
Floating origin durante um salto ou queda	A física continua; velocidade e direção são preservadas.
Carregar um save com semente diferente da atual	O mundo atual é descartado e o mundo do save é reconstruído.
Caverna sem saída ou sem entrada	A remoção de regiões pequenas e a ligação com a superfície evitam isso; se acontecer, a semente deve ser documentada como caso conhecido.
Inventário cheio	A coleta falha com aviso; o item permanece no chão.

Checkpoint

Ao fim da primeira semana, até [data do checkpoint], poste no Discord um vídeo curto ou GIF (até 1 minuto) mostrando o jogador andando pelo terreno com os 9 chunks aparecendo e sumindo, sem costura visível nas bordas. Não precisa ter caverna, itens nem inventário ainda.

O checkpoint existe porque a costura de borda e a coordenada global são onde a maioria trava, e é melhor descobrir isso com uma semana de folga.

Roteiro sugerido

1. Gere um chunk único com fBm, curva e faixas, e ande nele.
2. Divida em chunks com coordenada global e monte a grade 3x3. Verifique a costura. Poste o checkpoint.
3. Adicione o floating origin e teste longe da origem.
4. Gere a caverna em um grid separado e coloque-a no mundo.
5. Distribua itens por densidade, com chave, e implemente a coleta.
6. Modele o inventário e ligue à interface por eventos.
7. Implemente salvar e carregar.
8. Teste os estados da tabela, organize o projeto e grave o vídeo.

Entrega

Até [data de entrega], entregue:

- o repositório do projeto, executável e sem erros durante a demonstração;
- uma cena inicial pronta para jogar;
- um README curto com: motor e versão, controles, integrantes, e as decisões de modelagem: qual ruído e quais parâmetros, tamanho do chunk, como a chave do item é montada, como e quando o floating origin é disparado, como a caverna é gerada;
- um vídeo de até 5 minutos mostrando o terreno, a troca de chunks, a caverna, a coleta, o inventário e um salvar e carregar.

Critérios de conclusão

- ☐ O terreno é gerado por fBm com pelo menos 3 oitavas, curva de altura e 4 faixas.
- ☐ Os parâmetros do ruído são editáveis sem mudar código.
- ☐ Cada chunk é uma malha própria; há sempre 9 chunks carregados ao redor do jogador.
- ☐ Não há costura visível entre chunks.
- ☐ Um chunk que sai e volta é idêntico.
- ☐ O floating origin é disparado além de um limite e é invisível para o jogador.
- ☐ Existe pelo menos uma caverna por autômato celular (ou ruído 3D documentado), com entrada acessível.
- ☐ Itens nascem por mapa de densidade, só em faixas caminháveis, pousados no chão.
- ☐ Cada item tem chave determinística; coletados não reaparecem.
- ☐ O inventário tem modelo de dados separado da interface e atualiza por eventos.
- ☐ Salvar e carregar reconstroem o mesmo mundo com os itens coletados ausentes.
- ☐ Os estados da tabela são tratados.
- ☐ O checkpoint foi postado no Discord no prazo.
- ☐ README e vídeo entregues.