Helix Jump
Projeto Integrador VI

Caroline P. Deon[1], Emily L. Balestrin[2], Júlia Fernanda Levandoski[3], Suzana V. Deliberal[4]

[1]Universidade Regional Integrada do Alto Uruguai e das Missões

[2]Departmento de Engenharias e Ciência da Computação
Erechim -RS

103415@aluno.uricer.edu.br, 105140@aluno.uricer.edu.br

103534@aluno.uricer.edu.br, 103435@aluno.uricer.edu.br

1. Introdução

Os jogos digitais para dispositivos móveis têm ganhado cada vez mais destaque no cenário mundial. Dados da Newzoo (2023) mostram que o mercado *mobile* já é o maior dentro da indústria de *games*, movimentando bilhões de dólares por ano e crescendo de forma contínua. Fato esse que demonstra como os smartphones deixaram de ser apenas ferramentas de comunicação e se tornaram também uma das principais formas de entretenimento digital.

Ferramentas como a Unity Engine democratizaram o desenvolvimento de jogos *3D*, permitindo que estudantes e pequenos grupos criem projetos complexos com recursos avançados de Computação Gráfica e programação em *C#* (UNITY, 2023). No caso deste trabalho, é desenvolvido um jogo *3D* no estilo *Helix Stack* para Android, envolvendo física, colisões e sistema de pontuação.

Para apoiar a organização do grupo, são utilizadas ferramentas de suporte como Git/GitHub para versionamento de código e documentação, Trello para gestão de tarefas e a metodologia ágil Scrum, que favorece o desenvolvimento em ciclos iterativos (SCHWABER; SUTHERLAND, 2020). Além disso, cursos da Udemy servem como complemento para o aprendizado de Unity e programação, seguindo a tendência do ensino online como forma acessível de capacitação tecnológica (EDUCAUSE, 2022).

Dessa forma, este projeto busca integrar teoria e prática no desenvolvimento de um jogo *3D* para dispositivos móveis, proporcionando um aprendizado mais completo e colaborativo. Além de aplicar conceitos de Programação *Mobile*, Computação Gráfica e metodologias ágeis, o trabalho nos permite desenvolver habilidades de trabalho em equipe, planejamento e solução de problemas, que são essenciais tanto na vida acadêmica quanto no mercado de tecnologia. Ao combinar a experiência prática com o estudo de cursos e materiais de referência, ampliamos nossa compreensão e fortalecemos nosso conhecimento sobre o desenvolvimento de jogos digitais.

2. Ferramentas e tecnologias utilizadas

Para o desenvolvimento do jogo digital proposto, foi necessário o uso de diferentes ferramentas e tecnologias que apoiam a parte técnica. A escolha desses recursos levou em consideração aspectos como desempenho, acessibilidade, colaboração e alinhamento às práticas atuais da indústria de software.

2.1. Metodologia Scrum

O Scrum é uma metodologia ágil que se consolidou como uma das abordagens mais eficazes para o desenvolvimento de produtos em ambientes complexos e dinâmicos. Ele não é um processo prescritivo, mas um *framework* adaptativo que fornece diretrizes para gerenciamento de projetos com alta incerteza, priorizando a entrega contínua de valor e a colaboração entre equipes (Schwaber & Sutherland, 2020).

Baseado no Manifesto Ágil de 2001, o Scrum incorpora princípios como interações entre pessoas acima de processos rígidos, *software* funcional acima de documentação extensa, e adaptação constante às mudanças em vez de seguir um plano rígido (Agile Alliance, 2024). Esse modelo é sustentado por três pilares fundamentais: transparência, que garante visibilidade das metas e do progresso; inspeção, que possibilita avaliar o andamento do projeto regularmente; e adaptação, permitindo ajustes rápidos quando necessário.

A metodologia organiza o trabalho em ciclos iterativos chamados *Sprints*, geralmente com duração de uma a quatro semanas, nos quais é produzido um incremento funcional do produto. Cada ciclo envolve planejamento, execução, revisão e retrospectiva, promovendo um processo contínuo de melhoria. Essa característica torna o Scrum especialmente adequado para projetos que demandam flexibilidade, como desenvolvimento de *softwares*, aplicações móveis e jogos digitais, nos quais é necessário validar hipóteses e ajustar funcionalidades de forma ágil.

No contexto do desenvolvimento de jogos, como no caso deste projeto que visa criar um jogo *3D* para dispositivos Android utilizando a Unity Engine, a aplicação do Scrum é estratégica. Ela permite que mecânicas de jogabilidade, sistemas de física e interação, além de elementos visuais e interface, sejam implementados de forma incremental, testados a cada *Sprint* e adaptados de acordo com a experiência do usuário. Essa abordagem reduz riscos, aumenta a qualidade do produto final e promove maior integração entre programação, *design* e testes.

2.2. C#

O *C#* é uma linguagem de programação compilada, fortemente tipada e orientada a objetos, sendo a mais popular dentro do ecossistema .NET. Na maioria dos programas escritos em *C#*, utiliza-se o comando dotnet *build* para compilar os arquivos de origem em um pacote binário, e em seguida o comando dotnet *run* para executar o programa. O processo pode ser simplificado, já que o dotnet *run* realiza a compilação automaticamente quando necessário. Esses comandos fazem parte da interface de linha de comando do .NET (dotnet CLI), incluída no SDK do .NET, que fornece diversas ferramentas para criação, execução e modificação de projetos (MICROSOFT, 2025).

Trata-se de uma linguagem multiplataforma e de uso geral, permitindo que aplicações sejam desenvolvidas para diferentes dispositivos e sistemas operacionais, desde dispositivos móveis até servidores e soluções em nuvem. O *C#* combina os princípios da programação orientada a objetos com recursos da programação funcional e suporte a operações de baixo nível, possibilitando a criação de softwares de alto desempenho sem comprometer a segurança. Além disso, grande parte dos runtimes e bibliotecas do .NET são escritos em *C#*, e avanços na linguagem frequentemente beneficiam todos os desenvolvedores da plataforma (MICROSOFT, 2025).

Por ser fortemente tipada, cada variável, constante, parâmetro e valor de retorno em *C#* possui um tipo definido, garantindo que as operações realizadas no código sejam consistentes e seguras. A biblioteca de classes do .NET fornece tipos numéricos primitivos e também tipos complexos, como coleções, matrizes, manipulação de arquivos, conexões de rede e estruturas de data e hora. O compilador utiliza essas informações de tipo para validar as operações, de modo que, por exemplo, uma variável do tipo inteiro pode ser utilizada em operações aritméticas, enquanto uma variável booleana não pode. Esse mecanismo impede erros de execução e contribui para a confiabilidade do código (MICROSOFT, 2025).

2.3. Unity

De acordo com Tang (2019), a Unity fornece módulos de física, renderização, colisão e geração procedural que permitem ao desenvolvedor focar na lógica do jogo em vez da implementação de baixo nível de algoritmos gráficos. A engine também oferece suporte a dispositivos móveis (Android e iOS), o que a torna uma ferramenta adequada para projetos de jogos casuais do gênero arcade, como os baseados em mecânicas similares a *Helix Jump*.

O desenvolvimento de jogos digitais tridimensionais para dispositivos móveis exige o domínio de ferramentas de computação gráfica, motores de jogo (*game engines*) e paradigmas de programação aplicados à interação em tempo real. Nesse cenário, a Unity Engine se destaca como uma das soluções mais utilizadas no setor por oferecer suporte multiplataforma, integração com bibliotecas gráficas otimizadas e um modelo de desenvolvimento orientado a componentes, utilizando a linguagem *C#*.

A Unity Engine é uma das plataformas de desenvolvimento de jogos mais utilizadas no mundo, permitindo a criação de jogos 2D e *3D* para múltiplas plataformas, incluindo Android, iOS, Windows, consoles e realidade virtual (VR/AR) (UNITY, 2023). O ambiente integrado da Unity combina ferramentas de design, animação, física, iluminação e programação, permitindo que desenvolvedores construam experiências interativas de forma eficiente e escalável.

Um dos principais diferenciais da Unity é sua arquitetura baseada em componentes, que permite adicionar comportamentos específicos a objetos de jogo por meio de scripts em *C#* ou componentes nativos. Essa abordagem modular facilita a organização do código, a reutilização de recursos e a manutenção de projetos complexos, como o desenvolvimento de jogos *3D* com mecânicas de física e colisão (UNITY, 2023).

Além disso, a Unity oferece suporte a funcionalidades essenciais para dispositivos móveis, como entrada por toque, otimização de desempenho e gerenciamento de memória, garantindo que o jogo funcione de forma fluida em diferentes aparelhos Android (UNITY, 2023). Para acelerar o desenvolvimento, será possível utilizar recursos da Asset Store, como materiais, modelos *3D* e scripts de exemplo, reduzindo o tempo gasto na criação de elementos básicos do jogo (UNITY, 2023).

2.4. Trello

O Trello é uma aplicação *web* criada em 2011 pela empresa Fog Creek Software, atualmente pertencente à Atlassian. Segundo Atlassian (2021), o Trello é uma ferramenta

de gerenciamento de projetos baseada no método *Kanban*, que utiliza quadros, listas e cartões para organizar atividades de forma visual e colaborativa.

Conforme Anderson (2019), a principal vantagem do uso de ferramentas como o Trello está na visualização do fluxo de trabalho, permitindo que as equipes acompanhem o andamento das tarefas em tempo real, reduzindo falhas de comunicação e melhorando a produtividade. Cada quadro (*board*) pode representar um projeto, as listas (*lists*) indicam estágios do processo e os cartões (*cards*) representam tarefas específicas atribuídas a membros da equipe.

De acordo com Ferreira (2020), o uso de plataformas digitais de gerenciamento como o Trello contribui para a transparência na execução de atividades e para a gestão eficiente de equipes distribuídas, sendo amplamente utilizado tanto em contextos acadêmicos quanto profissionais.

No presente projeto, o Trello é adotado como recurso de apoio à gestão, permitindo a distribuição de responsabilidades entre os integrantes do grupo, o acompanhamento de prazos e a organização das etapas de desenvolvimento do jogo na Unity Engine. Dessa forma, a equipe pode alinhar tarefas de programação, modelagem e testes dentro de um mesmo ambiente digital, garantindo maior controle do progresso do trabalho.

2.5. Git e GitHub

O Git é um sistema de controle de versão distribuído, gratuito e de código aberto, desenvolvido inicialmente por Linus Torvalds em 2005, utilizado para registrar e gerenciar alterações em projetos de *software* de forma eficiente e segura. Diferente de sistemas centralizados, o Git permite que cada desenvolvedor possua uma cópia completa do repositório, com todo o histórico de alterações, garantindo independência, resiliência e rastreabilidade das modificações (GIT, 2025). Ele conta com comandos essenciais para criar repositórios, registrar alterações, sincronizar com repositórios remotos e controlar o histórico de versões, sendo amplamente flexível e adaptável a diferentes fluxos de trabalho (GIT, 2025).

O GitHub é uma plataforma que integra e amplia os recursos do Git, oferecendo uma interface web para gerenciamento de repositórios, controle de acessos, revisão de código por meio de *pull requests* e colaboração entre desenvolvedores. Além disso, disponibiliza recursos de automação como GitHub *Actions*, publicação de sites estáticos via GitHub *Pages* e *API's* para integração com ferramentas externas, garantindo maior eficiência e organização no desenvolvimento de projetos (GITHUB, 2025).

No contexto do desenvolvimento do jogo, o Git e o GitHub são utilizados para o versionamento do projeto, permitindo que todas as alterações no código, scripts, assets e recursos do jogo sejam organizadas e registradas de forma segura, facilitando o trabalho colaborativo da equipe e o acompanhamento do progresso do desenvolvimento.

3. Metodologia

O desenvolvimento do jogo *3D* estilo *Helix Jump* é realizado a partir de uma abordagem prática, integrando conceitos de programação, computação gráfica e metodologias ágeis. O processo é conduzido utilizando a engine Unity, escolhida por sua estabilidade e

suporte a projetos tridimensionais para dispositivos móveis. A linguagem de programação adotada é o *C#*, por estar integrada à Unity e oferecer recursos modernos para o controle de lógica, física e comportamento dos elementos do jogo.

Inicialmente, é definida a proposta do jogo e suas mecânicas principais, como a movimentação da esfera em queda, a rotação das plataformas, a detecção de colisões e o sistema de pontuação. Em seguida, será iniciada a construção do ambiente *3D*, com o uso do Visual Studio Code para o desenvolvimento dos scripts em *C#*, responsáveis pelo controle da jogabilidade.

Para apoiar a organização do trabalho, é adotada a metodologia ágil Scrum, estruturando o desenvolvimento em *sprints*. O Trello é utilizado para o gerenciamento de tarefas e acompanhamento do progresso, enquanto o GitHub serve como ferramenta de versionamento do projeto, permitindo o registro e a colaboração sobre as alterações de código e recursos.

Durante o andamento do projeto, serão realizados testes frequentes em dispositivos Android, a fim de validar as funcionalidades implementadas, garantindo fluidez, estabilidade e boa experiência para o usuário final. Ao término do desenvolvimento, pretende-se buscar *feedbacks* com crianças e adolescentes, público-alvo do jogo, permitindo avaliar o nível de diversão, acessibilidade e usabilidade da proposta. Essa etapa será fundamental para identificar melhorias e validar a experiência do jogador antes da entrega final.

4. Desenvolvimento

As mecânicas implementadas no jogo são projetadas para reproduzir a experiência característica de títulos do gênero *Helix Stack*, priorizando a simplicidade dos controles e a fluidez da jogabilidade. Os principais elementos desenvolvidos são: movimento da esfera, colisão com as plataformas, controle do jogo pelo usuário e sistema de pontuação.

4.1. Movimento da Esfera

O movimento da esfera baseia-se no sistema de física integrado da Unity Engine, que utiliza a biblioteca PhysX para simulações realistas. A esfera é configurada com um componente *Rigidbody*, responsável por aplicar as forças de gravidade e controlar a aceleração durante a queda. Assim, o deslocamento ocorre de forma natural, variando conforme as interações com as superfícies das plataformas. Como destaca Gregory (2019), a aplicação de motores físicos em game engines modernas é fundamental para garantir consistência e realismo nas interações entre objetos virtuais.

4.2. Colisão com Plataformas

A detecção de colisões é implementada por meio de componentes *Collider*, adicionados tanto à esfera quanto às plataformas. Cada colisão aciona eventos programados em scripts *C#*, responsáveis por identificar se a esfera atingiu uma superfície válida (plataforma segura) ou inválida (região de falha). Essa lógica é essencial para determinar o fluxo do jogo, influenciando ações como a continuidade da queda, a perda de pontos ou o reinício da fase.

4.3. Controle de Jogo

O sistema de controle é projetado para ser intuitivo e otimizado para dispositivos móveis. O jogador interage por meio de toques na tela, que rotacionam a torre de plataformas em torno do eixo vertical, permitindo que a esfera continue sua queda livre, cabendo ao usuário apenas abrir o caminho.

A lógica de rotação é implementada em *scripts C#*, utilizando a leitura das coordenadas de toque (*Touch Input*) e convertendo-as em movimentos angulares. Assim, gestos simples, como deslizar o dedo para a esquerda ou direita, resultam em movimentos suaves e precisos da estrutura. Conforme observa Schell (2020), a responsividade dos controles é determinante para a imersão do jogador, pois torna a interface quase "invisível" durante o processo de interação.

4.4. Dinâmica do Jogo

A dinâmica do jogo desenvolvido, inspirado em *Helix Jump*, segue um modelo de progressão em fases. O jogador controla a queda de uma esfera por meio da rotação de uma torre formada por plataformas circulares, como representado na subfigura 1(b). Cada fase apresenta um conjunto de plataformas que se tornam maiqs numerosas conforme o jogador avança.

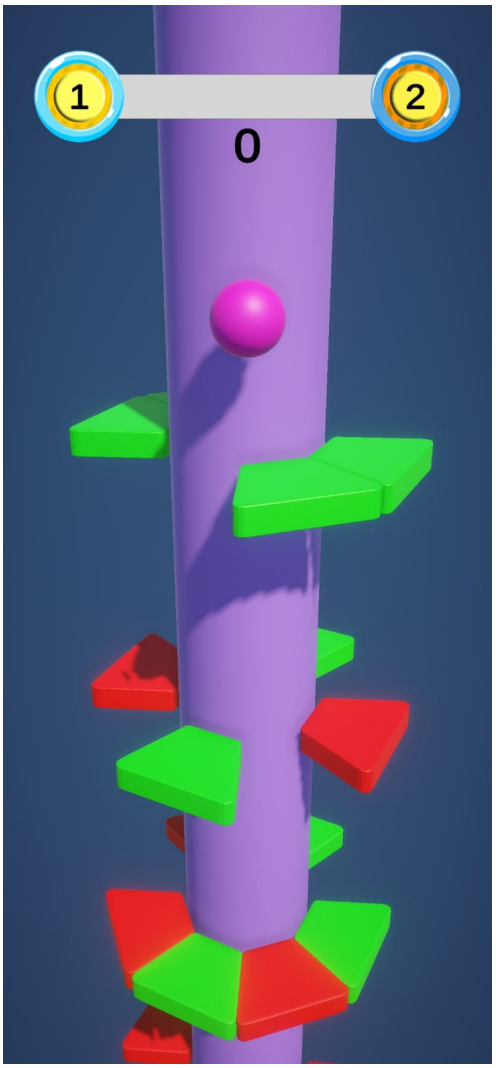
O objetivo é conduzir a esfera até a base da torre sem que ela colida com as áreas vermelhas, que representam zonas de perigo. Caso a colisão ocorra, a partida é encerrada e a tela da subfigura 2(a) é exibida. Já as áreas verdes correspondem às regiões seguras, permitindo que a esfera continue avançando pela fase.

A cada fase concluída, três novas plataformas são adicionadas, aumentando gradualmente o nível de dificuldade. O jogo também registra automaticamente a última fase alcançada, possibilitando que o jogador retome seu progresso na próxima execução.

O sistema de pontuação é baseado no número de plataformas superadas e no total de fases concluídas. Dessa forma, quanto maior o desempenho do jogador, maior será sua pontuação acumulada, incentivando a repetição e a busca por melhores resultados.

Além disso, o jogo conta com efeitos sonoros que reforçam a percepção das ações. O som do quique da esfera, por exemplo, contribui para a sensação de física realista, enquanto os efeitos associados à vitória ou derrota (como ilustrado nas subfiguras 2(b) e 2(a)) proporcionam um retorno auditivo imediato, favorecendo o engajamento e a imersão.

(a) Tela inicial do jogo



(b) Tela do jogo em execução

Figure 1. Telas iniciais e de gameplay.

(a) Tela exibida quando o jogador perde

(b) Tela exibida quando o jogador vence

Figure 2. Telas de término da fase.

4.5. Resultados Obtidos

O desenvolvimento do jogo *Helix Jump* resultou em um protótipo funcional que integra mecânicas de física, colisão, controle por toque e progressão por fases, compatível com dispositivos Android. A implementação permitiu validar tanto o funcionamento da lógica de jogo quanto a experiência do usuário em diferentes cenários de interação.

Entre os principais resultados, destaca-se a integração bem-sucedida do sistema de detecção de colisões, responsável por diferenciar áreas seguras (verde) e áreas de falha (vermelha), garantindo o fluxo básico da jogabilidade. O movimento da esfera, baseado na física da Unity, mostrou-se estável e responsivo, oferecendo uma sensação realista de queda e impacto. Os testes demonstraram que a interação por toque permite ao jogador controlar a rotação da torre de forma fluida, mantendo a jogabilidade simples e acessível.

O sistema de progressão em fases também foi concluído com êxito. A cada nível completado, o jogo gera automaticamente três novas plataformas, aumentando gradualmente o desafio. O progresso do jogador passa a ser salvo, permitindo a retomada do

jogo no ponto em que foi interrompido, funcionalidade importante para a experiência em dispositivos móveis.

Adicionalmente, foram incorporados efeitos sonoros que reforçam a imersão do jogador. O som de quique da esfera ao colidir com plataformas adiciona uma resposta tátil-auditiva à jogabilidade, enquanto os sons de vitória e derrota ajudam a comunicar visual e emocionalmente o resultado de cada tentativa.

Por fim, os testes preliminares indicam que o jogo apresenta boa estabilidade, desempenho adequado e potencial de evolução, com possibilidade de expansão para novas mecânicas, animações e melhorias visuais. O protótipo cumpre os objetivos definidos no início do projeto e constitui uma base sólida para versões futuras.

5. Conclusão

O desenvolvimento do jogo *Helix Jump* representou uma experiência significativa de integração entre teoria e prática, permitindo ao grupo aplicar conhecimentos adquiridos ao longo do curso em um projeto real de desenvolvimento de jogos. A construção de um jogo tridimensional, utilizando a Unity Engine e scripts em *C#*, possibilitou compreender de forma aprofundada conceitos de física, colisões, controle de objetos e organização de cenas internas à engine.

A utilização da linguagem *C#* foi essencial para estruturar a lógica do jogo e implementar comportamentos como rotação da torre, detecção de zonas de segurança e falha, progressão entre fases e sistema de pontuação. Esse processo contribuiu para o desenvolvimento de competências em programação orientada a objetos e no uso de funções e eventos aplicados ao contexto de jogos digitais.

Além dos aspectos técnicos, o projeto também proporcionou aprendizado sobre organização de equipe e divisão de responsabilidades. A aplicação da metodologia Scrum, juntamente com ferramentas como Trello e GitHub, facilitou a coordenação das atividades, o acompanhamento das entregas e o versionamento do código, garantindo maior controle sobre as etapas de desenvolvimento.

Outro elemento relevante foi o trabalho com *feedbacks* sonoros, que enriquecem a experiência do usuário e auxiliam na construção de respostas claras durante a jogabilidade. A implementação de sons de colisão, vitória e derrota evidenciou a importância do áudio como parte fundamental do engajamento em jogos casuais.

O protótipo final entregue está de acordo com os objetivos definidos inicialmente: o jogo funciona de forma estável tanto em dispositivos *mobile* quanto em *desktop*, neste último caso, exigindo apenas que o projeto seja executado pelo Unity Hub. Ele apresenta uma progressão coerente entre as fases, inclui elementos sonoros que reforçam a interação e conta com um sistema que registra a última fase alcançada pelo jogador. Esses resultados demonstram que o grupo conseguiu transformar um conceito teórico em um produto digital funcional. Além disso, toda a implementação, incluindo código-fonte e materiais de apoio, está disponível no repositório do GitHub, acessível em: **`https://github.com/EmilyBalestrin/Helix-Jump-Game`**.

Por fim, o desenvolvimento do jogo também mostrou que há várias possibilidades de continuidade. Entre elas, estão aprimorar o visual, adicionar desafios diferentes, permitir a personalização da esfera ou incluir pequenos efeitos e animações que deixem a

jogabilidade mais interessante. Mesmo sendo um protótipo, o projeto ajudou o grupo a aprender novas técnicas, a organizar melhor o trabalho e a colaborar de forma mais eficiente, o que contribuiu tanto para a prática quanto para o aprendizado ao longo da disciplina.

References

Agile Alliance (2025). What is scrum? Disponível em: https://www.agilealliance.org/agile101/scrum/.

Agile Manifesto (2025). Manifesto for agile software development. Disponível em: https://agilemanifesto.org/.

Anderson, D. J. (2019). *Kanban: Successful Evolutionary Change for Your Technology Business.* Sequim: Blue Hole Press.

Atlassian (2025). Trello. Disponível em: https://trello.com/. Acesso em: 22 ago. 2025.

Ferreira, J. C. (2020). *Gestão de projetos com ferramentas digitais: práticas e tendências.* São Paulo: Atlas.

Git (2025). Documentação do git. Git SCM. Disponível em: https://git-scm.com/docs/git. Acesso em: 22 ago. 2025.

GitHub (2025). Documentação do github. GitHub Docs. Disponível em: https://docs.github.com/pt. Acesso em: 22 ago. 2025.

Gregory, J. (2019). *Game Engine Architecture*. CRC Press, Boca Raton, 3 edition.

Guia do PC (2025). Relatório de tendências 2025 da udemy destaca uso da ia generativa para produtividade e capacitação profissional. Disponível em: https://www.guiadopc.com.br/pesquisas-estudos/53320. Acesso em: 30 set. 2025.

Microsoft (2025). Documentação do c#. Microsoft Learn. Disponível em: https://learn.microsoft.com/pt-br/dotnet/csharp/. Acesso em: 22 ago. 2025.

Neumann, M. and Baumann, L. (2021). Agile methods in higher education: Adapting and using eduscrum with real world projects. Disponível em: http://arxiv.org/abs/2106.12166.

Nystrom, R. (2014). *Game Programming Patterns*. Genever Benning, Nova Iorque, 1 edition.

Pressman, R. S. (2016). *Engenharia de Software: Uma Abordagem Profissional.* Porto Alegre: AMGH, 8th edition.

Rollings, A. and Adams, E. (2003). *Fundamentals of Game Design*. Prentice Hall, Upper Saddle River.

Schell, J. (2020). *The Art of Game Design: A Book of Lenses*. CRC Press, Boca Raton, 3 edition.

Schwaber, Ken.; Sutherland, J. (2020). The scrum guide. Disponível em: https://scrumguides.org/.

Schwaber, Ken.; Sutherland, J. (2025). Scrum guide. Disponível em: https://scrumguides.org/scrum-guide.html. Acesso em: 22 aug. 2025.

Sommerville, I. (2019). *Engenharia de Software: Uma Abordagem Profissional.* São Paulo: Pearson, 10th edition.

Swartout, S. (2020). Procedural generation in unity: Introduction to the art. Medium. Disponível em: https://medium.com/@simon.swartout/procedural-generation-in-unity-part-1-introduction-to-the-art-48deaeaaf44a. Acesso em: 22 ago. 2025.

TechTudo (2020). Udemy é confiável? veja como funciona site para fazer cursos online. Disponível em: https://www.techtudo.com.br/listas/2020/06/udemy-e-confiavel-veja-como-funciona-site-para-fazer-cursos-online.ghtml. Acesso em: 30 set. 2025.

Udemy (2010). Udemy company. Disponível em: https://about.udemy.com/company/. Acesso em: 22 ago. 2025.

Udemy (2025). Estudar na udemy: perguntas frequentes. Disponível em: https://support.udemy.com/hc/pt/articles/229232187-Estudar-na-Udemy-perguntas-frequentes?. Acesso em: 30 set. 2025.

Unity Technologies (2019). Unity user manual (2019.4 lts). San Francisco: Unity Technologies. Disponível em: https://docs.unity3d.com. Acesso em: 22 ago. 2025.

Unity Technologies (2023). Programming physics solutions for game development. Disponível em: https://unity.com/solutions/programming-physics. Acesso em: 22 ago. 2025.

Unity Technologies (2025). Unity 6.2 user manual. Disponível em: https://docs.unity3d.com/Manual/index.html. Acesso em: 12 sept. 2025.

Wijman, T. (2025). Last looks: The global games market in 2023. Disponível em: https://newzoo.com/resources/blog/last-looks-the-global-games-market-in-2023. Acesso em: 26 aug. 2025.

[Ferreira 2020], [Pressman 2016], [Sommerville 2019], [Anderson 2019], [**?**], [Microsoft 2025], [Git 2025], [GitHub 2025], [Schwaber 2020], [Agile Alliance 2025], [Agile Manifesto 2025], [Neumann and Baumann 2021], [Schwaber 2025], [Unity Technologies 2025], [Wijman 2025], [Udemy 2010], [Wijman 2025], [Unity Technologies 2019], [Unity Technologies 2023], [Swartout 2020], [Atlassian 2025], [Gregory 2019], [Schell 2020], [Rollings and Adams 2003], [Nystrom 2014], [Udemy 2025], [TechTudo 2020], [Guia do PC 2025]