Capítulo 3

Desenvolvimento de Jogos 2D de Plataforma: Explorando Unity e Chat GPT para Criação de Códigos Dinâmicos

Héder Pereira Rodrigues Silva, Carlos Heitor Marques da Silva Santos, Isabele Rodrigues de Souza, Artur Guilherme Silva Caldas, Iallen Gábio de Sousa Santos

Abstract

As technology advances, game development has become more accessible, allowing enthusiasts and independent developers to build their own virtual worlds. In addition to large companies, online tools like free-access game engines have democratized game development. In this course, we will focus on the Unity platform, a powerful game development engine for 2D and 3D games. We will explore how the artificial intelligence of Chat GPT can expedite the code creation process, freeing up time for creativity. We will discuss fundamental game development concepts, development tools, the C# programming language, and create a case study for developing a platform game.

Resumo

À medida que a tecnologia avança, a criação de jogos se tornou mais acessível, permitindo que entusiastas e desenvolvedores independentes construam seus próprios mundos virtuais. Além das empresas de grande porte, as ferramentas online, como as engines de acesso gratuito, democratizaram o desenvolvimento de jogos. Neste curso, focaremos na plataforma Unity, uma poderosa engine de desenvolvimento de jogos em 2D e 3D, e exploraremos como a inteligência artificial do Chat GPT pode acelerar o processo de criação de códigos, liberando tempo para a criatividade. Discutiremos os conceitos fundamentais do desenvolvimento de jogos, ferramentas de desenvolvimento, linguagem de programação C# e criaremos um estudo de caso para desenvolver um jogo de plataforma.

3.1. Introdução

Segundo o [SX group 2023], Pesquisa Game Brasil(PGB), de 2023, “Jogar jogos digitais é uma das mais importantes formas de diversão dentre os costumes de entretenimento brasileiro”. A diversão dos jogos está enraizada na variedade de emoções que eles podem evocar, desde a satisfação de superar obstáculos e resolver quebra-cabeças até a adrenalina da competição em tempo real. Os games oferecem uma variedade de experiências que despertam a imaginação, conhecimento e proporcionam entretenimento. Além disso, oferecem uma sensação de realização à medida que os jogadores progridem, ganham recompensas e alcançam objetivos dentro do jogo. Essa combinação de desafio, imersão e recompensa contribui para a diversão duradoura que os games proporcionam e torna esta forma de entretenimento cativante para pessoas de todas as idades.

O avanço da tecnologia, tem tornado a criação de seus seus próprios mundos virtuais mais acessível para entusiastas e desenvolvedores independentes. A 10ª edição da Pesquisa Game Brasil [SX group 2023], afirma no seu levantamento anual, visto na figura 3.1 consolidado sobre o consumo de jogos eletrônicos no país, que cerca de 70,1% afirmam ter o hábito de jogarem, jogos eletrônicos. O desenvolvimento de jogos eletrônicos tem algumas particularidades em relação ao desenvolvimento tradicional de software. Este fator é intensificado pela integração com o fator artistico impermeado neste meio.

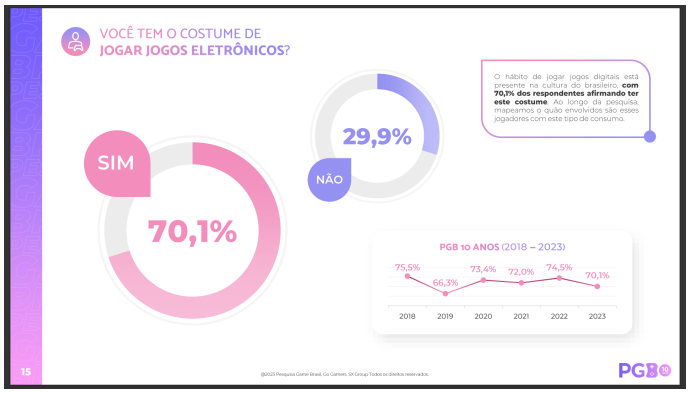


Figura 3.1. Consumo dos jogos eletrônicos, PGB 10 anos. [SX group 2023]

Os jogos desempenham um papel de relevância, abrangendo uma ampla gama de áreas como: educação, entretenimento, saúde e aplicações profissionais. Na educação, eles podem ser usados em salas de aula e atividades extracurriculares para tornar o aprendizado mais envolvente. Na saúde, os jogos têm sido aplicados para melhorar a atividade física e auxiliar no tratamento de distúrbios mentais. No campo profissional, servem como ferramenta de treinamento e até mesmo como estratégia de marketing. Finalmente, o entretenimento é uma das principais aplicações dos jogos, abrangendo desde passatempos casuais até competições globais online e experiências imersivas com narrativas envolventes.

Os videogames abrangem uma vasta variedade de gêneros e estilos, cada um com suas próprias características e desafios distintos. Por exemplo, jogos de estratégia desafiam a mente enquanto os jogos de tiro em primeira pessoa testam os reflexos do jogador. O mundo dos jogos oferece algo para todos os gostos.

Tendo em vista isso, os jogos de plataforma são destacados neste capitulo, uma vez que geralmente apresentam controles simples, com movimentos básicos, como pular e correr. Isso os torna acessíveis a uma ampla gama de jogadores, incluindo iniciantes e jogadores casuais. Mas fora isso, você basicamente se deslocava pela fase, coletava itens e pulava em blocos ou desviava de obstáculos. Ainda assim, à medida que o tempo progrediu, novas aptidões e ferramentas começaram a surgir, evoluindo desde a pré-renderização (também chamado 2,5d) até o total. Alguns jogos de plataforma 2d entraram para a história alcançando sucesso comercial e um grande número de fãs dando origem a franquias que duram até hoje. Entre eles podem ser destacados: Sonic[SEGA 1991], Mario [Nintendo 1983], Mega Man [Nintendo 1987], como visto na Figura 3.2 e tantos outros. Estes jogos são marcados por jogabilidade simples e desafiadora.

Figura 3.2. Cena de Batalha do jogo Mega Man. [Nintendo 1987]

A simplicidade pode ser o segredo para criar jogos divertidos e com uma elevada taxa de repetição. O ponto a considerar é que os jogos de plataforma estão presentes, mesmo após décadas desde o seu surgimento, e novos títulos continuam a ser lançados ano após ano, cativando a preferência do público, o que demonstra que esse gênero ainda é atraente e que uma fórmula pode ser ajustada, mantendo a mesma essência original, sem se tornar obsoleta. Adicionalmente, a indústria de jogos se transformou em uma potência financeira, movimentando bilhões de dólares e gerando oportunidades empolgantes de carreira para desenvolvedores talentosos.

Desde que emergiram, há algumas décadas, os videogames representam diversão. No entanto, para além daquele público jovem que historicamente se vinculou a essa forma de entretenimento, existe hoje uma audiência adulta que também está envolvida nesse passatempo. Nesse sentido, tal setor gera aproximadamente US$ 200 bilhões por ano, conforme estimativas da Newzoo, plataforma de informações sobre o universo dos jogos. [Michelle Rouhof 2023]. O desenvolvimento de jogos não é mais apenas uma forma de entretenimento; tornou-se um campo lucrativo e multifacetado, com aplicações que se estendem por uma ampla gama de áreas, incluindo educação, saúde e entretenimento.

No cerne desse crescimento, destacam-se as empresas que desempenham um papel vital na evolução da indústria. No ano fiscal de 2020/2021, a Sony, uma das empresas

Tabela 3.1. Ranking de empresas no mercado dos jogos e seus respectivos valores em outubro de 2023.

Empresa	Valor de mercado
Apple	US$ 3 Trilhões
Microsoft	US$ 2,63 Trilhões
Alphabet	US$ 1,53 Trilhão
Amazon	US$ 1,35 Trilhão
Meta, Inc.	US$ 780,2 Bilhões

notáveis na indústria de jogos, bateu recordes de receita e lucro na divisão de Games e Serviços de Internet, que inclui a PlayStation. De acordo com o seu último relatório financeiro, a empresa arrecadou impressionantes US$ 25,04 bilhões e teve um ganho de US$ 3,22 bilhões nessa área[Sony 2021]. Esses números destacam a robustez do mercado de jogos e a importância das grandes empresas na condução desse setor em constante crescimento. Suas vastas operações abrangem uma ampla variedade de gêneros e plataformas, abrindo oportunidades para uma indústria que continua a florescer.

A Tabela 3.1 apresenta um ranking de algumas das principais empresas no mercado de tecnologia e jogos, acompanhadas de seus respectivos valores de mercado em bilhões de dólares americanos, especificamente relacionados aos lucros obtidos no segmento de jogos dessas indústrias. A Apple lidera a lista com um valor de mercado de US$ 3 trilhões, seguida pela Microsoft com US$ 2,63 trilhões, a Alphabet (empresa-mãe da Google) com US$ 1,53 trilhão e a Amazon com US$ 1,35 trilhão. Além dessas quatro gigantes, a Meta, Inc. também está presente na lista, com um valor de mercado de US$ 780,2 bilhões.

A tabela apresenta uma coluna mostrando

O notável crescimento da indústria de jogos é um testemunho não apenas da sua capacidade de gerar lucros significativos para as empresas envolvidas, mas também da sua versatilidade e capacidade de impactar positivamente várias áreas da sociedade. No entanto, não são apenas as grandes corporações que se beneficiam desse crescimento.

A disponibilidade de ferramentas online, como as engines de acesso privado e gratuitas, que oferecem um conjunto de recursos e ferramentas para simplificar tarefas como gráficos, física e interatividade no desenvolvimento e execução de jogos de vídeo, democratizou o desenvolvimento de jogos, tornando-o acessível a um público mais amplo. A exemplo disso, temos plataformas como Unity [Unity Technologies 2023], Game Maker [Mark Overmars 2023], Unreal Engine[Epic Games 2023] e outras têm desempenhado um papel crucial nessa revolução. Elas tornaram a criação de jogos de alta qualidade mais acessível do que nunca, estimulando a inovação e a diversidade no setor. No entanto, neste minicurso, concentramos nossa atenção especificamente na Unity com o auxílio do Chat GPT, explorando como essas ferramentas podem colaborar para criar jogos dinâmicos de plataforma.

A Unity, em particular, destaca-se como uma plataforma de desenvolvimento de jogos e aplicativos em 2D e 3D. Ela oferece um ambiente de criação flexível e intuitivo, permitindo que programadores e artistas colaborem na criação de experiências interativas

para várias plataformas, incluindo PC, consoles, dispositivos móveis e realidade virtual. Com recursos avançados em gráficos, física realista, suporte a várias linguagens de programação e uma loja de ativos, a Unity é uma escolha popular para desenvolvedores em todo o mundo.

O Chat GPT, ou *Generative Pre-trained Transformer for Chat*, representa uma inovação significativa em inteligência artificial. Treinado em grandes volumes de texto, esse modelo de linguagem é capaz de compreender o contexto das conversas e gerar respostas coerentes e relevantes, tornando-o ideal para aplicações de chatbots, assistentes virtuais e enriquecimento da narrativa em jogos. Sua versatilidade e capacidade de entender a linguagem natural o tornam uma ferramenta valiosa em uma ampla variedade de campos, impulsionando a interatividade e a automação de tarefas relacionadas à geração de texto.

Neste minicurso, exploraremos como utilizar o Chat GPT como uma ferramenta para a criação de códigos, simplificando e acelerando o processo de desenvolvimento de jogos. pois a AI automatiza tarefas complexas, gerando códigos personalizados com base nas necessidades do desenvolvimento, assim, liberando tempo e recursos para se concentrar na criatividade e na jogabilidade.

Em conclusão, o notável crescimento da indústria de jogos é um testemunho não apenas da sua capacidade de gerar lucros significativos para as empresas envolvidas, mas também da sua versatilidade e capacidade de impactar positivamente várias áreas da sociedade, e ferramentas de desenvolvimento acessíveis. Com isso, Nos próximos capítulos, abordaremos os conceitos básicos para o desenvolvimento de um jogo, após isso uma visão sobre as ferramentas de desenvolvimento no mercado, então será apresentado o basico da linguagem C# que será utilizada e por fim um estudo de caso da criação de um jogo de plataformal.

3.2. Conceitos Básicos para o Desenvolvimento de Jogos

Diferentemente de outras áreas do desenvolvimento de software, a criação de jogos envolve conceitos específicos e peculiares. Neste capítulo, iremos explorar os fundamentos que servem como alicerce para a construção de videogames. É importante compreender os princípios que definem a essência de um jogo, incluindo a análise das regras que governam seu funcionamento, os objetivos que orientam os jogadores e os desafios que os mantêm imersos na experiência.

Considerando que os videogames são predominantemente utilizados para fins de entretenimento, os aspectos artísticos desempenham um papel essencial na criação de um jogo, incluindo elementos como trilha sonora e artes em 2D ou 3D. Estes aspectos contribuem para a composição de cenários, fases, personagens e outros elementos.

3.2.1. Mecânicas de jogo e Regras

Mecânicas de jogos, também conhecidas como mecânicas de gameplay, são sistemas e regras que definem como um jogo funciona do ponto de vista da interação do jogador. Elas estabelecem as ações que os jogadores podem realizar, as regras que regem essas ações e como essas ações afetam o mundo virtual do jogo. As mecânicas de jogos são

essenciais para criar desafios e os objetivos de um jogo. Aqui estão alguns exemplos das diferentes formas de mecânicas:

- **Definição de Ações Permitidas e Proibidas:** As mecânicas determinam quais ações os jogadores podem ou não realizar durante o jogo. Por exemplo, em um jogo de xadrez, as regras estipulam como cada peça pode se mover e as condições sob as quais uma peça pode ser capturada.
- **Estrutura de Turnos ou Fases:** Uma mecânica amplamente adotada é o sistema de batalhas em turnos, como pode ser visto na figura 3.3. Nesse sistema, os jogadores têm a oportunidade de escolher entre várias opções, como atacar, defender ou usar itens durante sua vez. Após a execução de suas ações, a vez passa para os inimigos, seguindo essa alternância. Esse sistema coloca ênfase no pensamento estratégico e no planejamento das jogadas, ao contrário da agilidade requerida em jogos de plataforma, por exemplo.

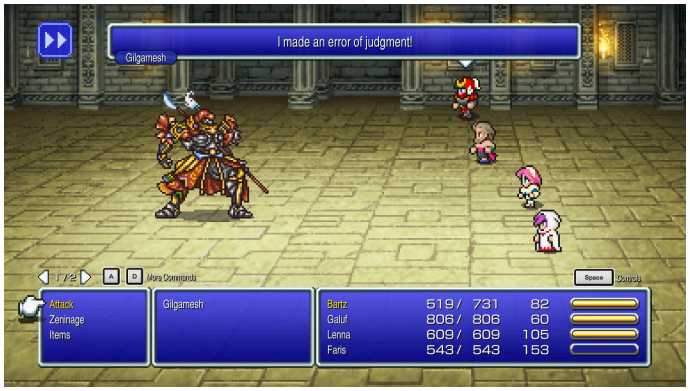


Figura 3.3. Cena de Batalha do jogo Final Fantasy V. [SquareSoft 1992]

- **Condições de Vitória e Derrota:** As regras definem as condições que determinam quando um jogador vence ou perde o jogo. Por exemplo, no jogo de tabuleiro "Catan", as regras estipulam que o primeiro jogador a atingir um certo número de pontos vence.
- **Limites de Recursos e Habilidades:** Muitos jogos têm recursos limitados ou habilidades específicas que os jogadores podem usar. As regras estabelecem como esses recursos são distribuídos e como as habilidades podem ser usadas. Por exemplo, em um jogo de estratégia em tempo real como "StarCraft II"[Entertainment 2010], as regras determinam a quantidade de recursos que os jogadores podem extrair e como as unidades podem ser treinadas e controladas.
- **Movimento:** A grande maioria dos jogos possui ou tem como base o controle de movimento de um personagem, assim, essa mecânica é explorada de varias formas diferentes. Por exemplo, no jogo "Sonic the Hedgehog"[SEGA 1991] o movimento consiste em correr em altas velocidades, enquanto no jogo "Celeste" [Thorson 2018] o foco é em pulos precisos da personagem e na escalada de paredes.

Em resumo, as regras e mecânicas de um jogo trabalham em conjunto para criar uma estrutura que dá forma à jogabilidade e à experiência geral do jogador. Elas estabelecem as diretrizes gerais e detalham como as ações dos jogadores se desenrolam e se conectam para criar a experiência de jogo desejada.

3.2.2. Objetivos

Os objetivos em jogos são as metas que os jogadores buscam alcançar durante a sua experiência de jogo. Essas metas dão propósito à jogabilidade, fornecendo aos jogadores um motivo para se envolverem e progredirem. Estes objetivos podem vir das mais diferentes formas, como por exemplo:

- **Motivação e Direcionamento:** Os objetivos fornecem motivação aos jogadores, dando-lhes um senso de propósito e direção. Por exemplo, em um jogo de aventura como "The Legend of Zelda"[Nintendo 1986], o objetivo principal é resgatar a princesa e derrotar o vilão, proporcionando um propósito claro para o jogador e uma direção para seguir em sua aventura.
- **Variedade de Objetivos:** Os objetivos podem variar amplamente, desde simples tarefas até conquistas complexas. Alguns jogos podem ter múltiplos objetivos ao longo da história. Em um jogo de simulação como "The Sims"[Arts 2000], os objetivos vão desde realizar interações sociais até conquistar metas de carreira e construir uma casa ideal.
- **Recompensas e Reconhecimento:** Ao atingir objetivos, os jogadores frequentemente são recompensados com pontos, itens, poderes especiais ou reconhecimento dentro do jogo. Em um jogo de RPG como "The Elder Scrolls V: Skyrim"[Bethesda 20], os objetivos de missões concedem recompensas valiosas, como equipamentos poderosos e habilidades adicionais. Assim servindo como motivação adicional para o jogador continuar jogando.

Em resumo, os objetivos em jogos são elementos cruciais para manter os jogadores engajados, proporcionar desafios e criar uma sensação de progresso. Eles variam de acordo com o tipo de jogo e podem ser personalizados para atender às necessidades específicas da experiência de jogo desejada.

3.2.3. Desafios

Os desafios desempenham um papel fundamental para tornar a experiência de jogo mais envolvente para os jogadores. Eles são essenciais para enriquecer a jornada em direção aos objetivos, tornando-a mais significativa e recompensadora.

Os desafios, muitas vezes, proporcionam um senso de realização à medida que os jogadores superam obstáculos. Eles podem testar as habilidades e conhecimentos dos jogadores, encorajando-os a desenvolver estratégias e táticas criativas para vencer. Essa superação progressiva dos desafios cria uma sensação de crescimento pessoal dentro do jogo à medida que os jogadores adquirem novas habilidades.

Além disso, os desafios também contribuem para a imprevisibilidade e variedade em jogos. Eles quebram a monotonia, garantindo que cada sessão de jogo seja única.

A incerteza de como um desafio será enfrentado ou resolvido adiciona um elemento de surpresa e emoção à experiência.

No entanto, é crucial que os desenvolvedores mantenham um equilíbrio cuidadoso ao criar desafios. Desafios excessivamente difíceis podem levar a uma experiencia frustrante, enquanto desafios muito fáceis resultam em tédio. Portanto, encontrar o ponto ideal de dificuldade para o jogo é o essencial, para assim manter os jogadores imersos e ansiosos por mais.

aqui estão alguns exemplos de como os desafios são aplicados em diferentes tipos de jogos e situações:

- **Jogos de Plataforma:** Em jogos como "Super Mario"[Nintendo 1983], os desafios frequentemente envolvem a navegação por níveis cheios de obstáculos, como buracos, inimigos e armadilhas, enquanto o jogador tenta alcançar a bandeira no final.
- **Quebra-Cabeças:** Jogos de quebra-cabeças, como "Tetris"[Pajitnov 1984] ou "Candy Crush Saga"[King 2012], desafiam os jogadores a pensar rapidamente e tomar decisões estratégicas para criar combinações ou encaixar peças em padrões específicos.
- **Jogos de RPG:** Em jogos de RPG como "The Witcher 3: Wild Hunt"[RED 2015], os desafios incluem batalhas contra inimigos formidáveis, que requerem planejamento tático, escolha de habilidades e uso de poções para vencer.
- **Jogos de Estratégia em Tempo Real (RTS):** Jogos como "StarCraft II"[Entertainment 2010] apresentam desafios que envolvem o gerenciamento de recursos, construção de bases e coordenação de unidades em batalha, exigindo habilidades estratégicas e táticas.
- **Jogos de Sobrevivência:** Em jogos de sobrevivência como "Minecraft"[Mojang 2011], os desafios incluem encontrar recursos, enfrentar monstros e construir abrigos para sobreviver em um ambiente hostil.

3.3. Ferramentas para o Desenvolvimento de Jogos

Para a criação e desenvolvimento de um jogo, são necessários diversos tipos de software e ferramentas. Essas ferramentas desempenham um papel fundamental, tanto na criação e edição de códigos para os algoritmos que operam dentro do jogo, quanto na criação dos elementos audiovisuais que compõem o videogame. Nesta seção, exploraremos alguns exemplos de recursos amplamente utilizados por desenvolvedores, desde iniciantes até os mais experientes.

3.3.1. Engines de jogos

Uma engine de jogos é uma plataforma de desenvolvimento que fornece um conjunto de ferramentas e recursos para criar jogos de forma mais eficiente. Ela inclui componentes essenciais, como mecanismos de física, renderização, áudio e lógica de jogo, permitindo que os desenvolvedores foquem na criação de conteúdo. As engines de jogos são amplamente utilizadas na indústria para acelerar o processo de desenvolvimento e oferecer

suporte a uma variedade de plataformas. Abaixo, apresentaremos algumas das engines de jogos mais conhecidas e suas características distintas.

- **Unity** é uma das engines de desenvolvimento de jogos mais populares e amplamente utilizadas no mundo. Ela é conhecida por sua versatilidade e suporte multiplataforma, o que a torna uma escolha ideal para desenvolvedores que desejam criar jogos para uma variedade de dispositivos, incluindo PC, consoles, dispositivos móveis e realidade virtual.
- **Gamemaker** é uma engine de desenvolvimento de jogos que ganhou destaque por sua abordagem acessível e amigável para iniciantes. Ela permite que desenvolvedores criem jogos 2D de forma rápida e eficaz, usando uma linguagem de programação chamada GameMaker Language (GML) ou uma interface de arrastar e soltar.
- **Godot Engine** é uma engine de código aberto que ganhou popularidade devido à sua simplicidade e facilidade de uso. É uma opção excelente para desenvolvedores iniciantes, mas também é poderosa o suficiente para projetos avançados. Godot possui seu próprio sistema de script chamado GDScript, que é semelhante a Python.

3.3.2. Artes e Música

Como mencionado na Seção 3.1, os elementos gráficos e sonoros desempenham um papel fundamental nos jogos. A trilha sonora, por exemplo, tem o poder de evocar emoções, criar atmosferas de mundo e fortalecer a narrativa e o ambiente do jogo. Por outro lado, as artes visuais dão vida ao mundo virtual, enriquecendo-o e tornando-o mais imersivo para os jogadores. A ausência desses elementos resultaria em experiências monótonas e desinteressantes. Portanto, o desenvolvimendo de videogames requer habilidades artísticas bem como o domínio de ferramentas para produção destes elementos. A seguir, são apresentadas algumas ferramentas que podem ser utilizadas para o a criação desses elementos.

- **Softwares para música:**
  - **Ableton Live:** O Ableton Live é um software de produção musical amplamente usado tanto em estúdio quanto em performances ao vivo Ele oferece uma ampla gama de instrumentos virtuais e efeitos em tempo real, assim também sendo eficiente para a criação de trilha sonora e efeitos de ambiente.
  - **FL Studio:** também conhecido como Fruity Loops, é um software de produção musical versátil apreciado por sua interface amigável. É uma escolha popular para iniciantes e produtores de música eletrônica devido à sua flexibilidade.
- **Softwares para desenho 2d/3d**
  - **Adobe Photoshop:** O Photoshop é uma ferramenta versátil amplamente utilizada para a criação de artes 2D. Ele oferece recursos avançados de edição de imagem, pintura digital e ilustração. Pode ser usado para criar personagens, cenários e elementos visuais em vários estilos artísticos, desde realismo até ilustrações estilizadas e pixel art.

– **Piskel:** O Piskel é um software focado em pixel art, tornando-o uma escolha ideal para artistas que desejam criar gráficos pixelados, comuns em jogos retrô e indie. Ele possui uma interface simples e ferramentas específicas para desenhar e animar sprites em pixel art.
– **Blender:**O Blender é uma ferramenta de modelagem e animação 3D. Embora seja mais conhecido pela criação de modelos tridimensionais, também é usado para criar ativos 3D para jogos, como personagens, objetos e ambientes. Oferece recursos avançados de modelagem, texturização e animação.

Por fim, tendo em vista as ferramentas demonstas, neste minicurso estaremos focando na plataforma Unity como a engine de desenvolvimento de jogos principal para nossos projetos práticos. Além disso, para auxiliar na criação de códigos e fornecer suporte adicional, também utilizaremos o chat GPT.

3.4. Sintaxe Básica de C#

O C# (pronunciado como "C sharp") é uma linguagem de programação versátil e amplamente usada desenvolvida pela microsoft. Uma vez que este minicurso se propõe a introduzir o desenvolvimento de jogos com a Unity, é importante conhecer a sintaxe básica de C# pois esta é a linguagem padrão de desenvolvimento nesta Engine.

As sintaxes são as regras que governam a estrutura gramatical e a forma correta de escrever código. À medida que exploramos esses conceitos básicos de sintaxe, também mencionaremos como eles são usados no contexto da programação com unity, oferecendo uma visão geral dos fundamentos necessários para começar o desenvolvimento de jogos.

3.4.1. Estrutura basica de C#

Todo programa C# começa com uma estrutura básica, essa estrura é a base pra cirar programas em C#. Na Unity, esta estrutura é criada automaticamente e não é alterada ou gerenciada pelo desenvolvedor de maneira direta. O Algoritmo 3.1 apresenta esta estrutura básica de um programa em C#.

Algoritmo 3.1. Estrutura básica de C#

```
1 using System;
2 class program {
3           static void Main(){
4          //Seu codigo aqui
5         }
6 }
```

- "USING SYSTEM"; isso inclui o namespace "SYSTEM", que contém classes e métodos essenciais.
- "CLASS PROGRAM’; declara uma classe chamada "PROGRAM".
- "STATIC VOID MAIN ()"; O ponto de entrada. Todo programa C# começa a ser executado a partir deste metodo.

3.4.2. Váriaveis e tipos de dado

Uma variável em C# permite que você organize e manipule valores, como números, textos ou datas, durante a execução do programa. Os tipos de dados definem o tipo de valor que uma variável pode conter. No Algoritmo 1.2. são apresentados exemplos de declaração de variável em C#.

Algoritmo 3.2. variaveis e tipos de dado

```
int age = 25;
double salary = 1000.50;
string name = "Joao";
bool isactive = true;
```

- "INT"; Tipos de dados para números inteiros.
- "STRING"; Tipos de dados para textos.
- "FLOAT"; Tipos de dados para números de pontos flutuante.
- "BOOL"; Tipos de dados para valores booleanos (verdadeiros ou falsos).

3.4.3. Estruturas Condicionais

As estruturas condicionais em C# permite que você tome decisões em seu programa com base em condições. Ela controla qual parte do código será executada com base em se uma condição é verdadeira ou falsa. O bloco mais comum de estrutura condicional em C# é o if. O algoritmo 3.3 exemplifica a estrutura condicional if/else em C#.

Algoritmo 3.3. Estrutura condicionais

```
if (idade >= 18){
    Console.WriteLine(nome + "E maior de idade.");
}else{
    Console.WriteLine(nome + "E menor de idade.");
}
```

- "IF (idade >= 18)": Aqui, você verifica se a idade é maior ou igual a 18.
- ’ Console.WriteLine(nome + "é maior de idade.")’; : Se a condição do if for verdadeira, esta linha será executada e imprimirá no console "João é maior de idade."
- "ELSE": Caso a condição do if seja falsa, o código dentro do bloco else será executado.
- Console.WriteLine(nome + "é menor de idade."); : Se a condição do if for falsa, esta linha será executada e imprimirá no console "João é menor de idade."

3.4.4. Estruturas de repetição

As estruturas de repetição em C# (e em outras linguagens de programação) são usadas para executar um bloco de código várias vezes. Isso é útil quando você precisa realizar tarefas repetitivas ou interar sobre uma coleção de dados. Existem várias estruturas de repetição em C#, dentre elas, "*For*"e "*while*",como visto no algoritmo 3.4

Algoritmo 3.4. Estrutura de repetição

```
for (int i = 0; i < 5; i++){
    // Codigo a ser repetido
}
while (condicao){
    // Codigo a ser repetido enquanto a condicao for
       verdadeira
}
```

- “FOR”: Usado para criar um loop com um computador.
- “WHILE”: Cria um loop enquanto uma condição for verdadeira.

3.4.5. Arrays

Um array em c# é uma estrutura de dados que armazena uma coleção de elementos do mesmo tipo em uma sequência unidimensional. Esses elementos são acessados por meio de índices numéricos, onde cada elemento tem um índice que representa sua posição na sequência. O primeiro elemento de um array geralmente tem um índice de 0, o segundo tem um índice de 1, o terceiro tem um índice de 2 e assim por diante, seguindo uma sequência numérica crescente.

Algoritmo 3.5. Arrays

```
int[] numeros = new int[5];
numeros[0] = 10;
numeros[1] = 20;
numeros[2] = 30;
numeros[3] = 40;
numeros[4] = 50;
foreach (int numero in numeros){
    Console.WriteLine(numero);
}
```

- "INT [] NUMEROS": declara um array de inteiros chamado numeros com espaço para 5 elementos.
- Os valores são atribuídos aos elementos individuais do array usando índices, como "NUMEROS [0] = 10", "NUMEROS [1] = 20", etc.
- O loop "FOREACH" é usado para interar pelos elementos do array e imprimir cada valor.

3.4.6. Classes, Instâncias, Atributos, Métodos e Herença

As classes são uma parte fundamental da Programação Orientada a Objetos (POO), que é um paradigma de programação que organiza o código em torno de objetos e suas interações. As classes permitem a encapsulação de dados e funcionalidades relacionados, facilitando a reutilização do código e o gerenciamento de complexidade em projetos de software.

Algoritmo 3.6. Classe e POO

```
// Exemplo de declaracao de classe simples
class Pessoa{
    // Campos (variaveis de instancia)
    public string Nome;
    public int Idade;
    // Metodo construtor
    public Pessoa(string nome, int idade){
        Nome = nome;
        Idade = idade;
    }
    // Metodo de exemplo
    public void Apresentar(){
        Console.WriteLine($"Ola, meu nome e {Nome} e tenho
            {Idade} anos.");
    }
}
```

3.4.6.1. Objeto (Instancias)

Um objeto é uma instância de uma classe. Cada objeto possui seu próprio conjunto de campos e pode chamar os métodos da classe.

Algoritmo 3.7. Objeto da Classe Pessoa

```
// Criando objetos da classe Pessoa
Pessoa pessoa1 = new Pessoa("Alice", 30);
Pessoa pessoa2 = new Pessoa("Bob", 25);
// Chamando o metodo Apresentar para cada objeto
pessoa1.Apresentar(); // Saida: Ola, meu nome e Alice e
   tenho 30 anos.
pessoa2.Apresentar(); // Saida: Ola, meu nome e Bob e tenho
    25 anos.
```

- Objeto da Classe Pessoa: 'PESSOA1' é uma instância (objeto) da classe 'PESSOA'.

3.4.6.2. Atributos

Os atributos, também conhecidos como campos, são variáveis que pertencem a uma classe e representam o estado ou as características dos objetos dessa classe, como foi visto no código de classe. 3.6

3.4.6.3. Métodos

Os métodos são funções que pertencem a uma classe e definem o comportamento ou as ações que os objetos dessa classe podem realizar.

Algoritmo 3.8. Métodos da Classe Calculadora

```
class Calculadora{
    // Metodo que adiciona dois numeros inteiros
    public int Somar(int a, int b){
        return a + b;
    }
}
```

- Neste exemplo, criamos um objeto 'CALCULADORA' da classe 'CALCULADORA' e chamamos seu método 'SOMAR()' para calcular a soma de dois números.
- 'SOMAR(INT a, INT b)' é um método público da classe 'CALCULADORA' que aceita dois argumentos (números inteiros 'A' e 'B') e retorna a soma deles.

3.4.6.4. Herança

A herança é um conceito fundamental na programação orientada a objetos (POO) que permite que uma classe (chamada de classe derivada ou subclasse) herde os atributos e métodos de outra classe (chamada de classe base ou superclasse). Isso promove a reutilização de código e estabelece uma relação hierárquica entre as classes.

Algoritmo 3.9. Herança da Classe "Animal"

```
class Animal{
    public void Respirar(){
        Console.WriteLine("O animal esta respirando.");
    }
}

class Cachorro : Animal{
    public void Latir(){
        Console.WriteLine("O cachorro esta latindo.");
    }
}
```

- Neste exemplo, a classe 'CACHORRO' herda da classe 'ANIMAL'. Isso significa que um objeto da classe 'CACHORRO' terá acesso ao método 'EMITIR SOM'() da classe 'ANIMAL'. Além disso, a classe Cachorro tem seu próprio método 'LATIR'() e substitui (sobrescreve) o método 'EMITIR SOM'(), fornecendo uma implementação específica para cães.

3.5. Estudo de Caso: Desenvolvendo seu primeiro jogo com o auxílio da unity e chat gpt.

Como citado anteriormente, a Unity é uma engine de game design utilizada por várias empresas e desenvolvedores independentes. Além disso, a unity possui e disponibiliza milhares de recursos e tutoriais gratuitos aos desenvolvedores. Utilizando-a em conjunto com o Chat GPT, inteligência artificial mundialmente conhecida por sua facilidade de compreensão e possibilidade de recursos disponibilizados, é possível a criação de milhões de jogos digitais.

3.5.1. Preparação do Ambiente de Desenvolvimento

Nesta seção, serão abordados os preparos necessários para criação do jogo.

Inicialmente, para o desenvolvimento de jogos utilizando a Unity & ChatGPT, é necessário fazer o download da plataforma de desenvolvimento unity, acessando o site oficial da plataforma [Unity Technologies 2023] Com a Unity Hub instalada, é necessário fazer o download do software de desenvolvimento. De acordo com o site oficial da Unity [Unity Technologies 2023], os requisitos de sistema mínimos para utilização da unity são:

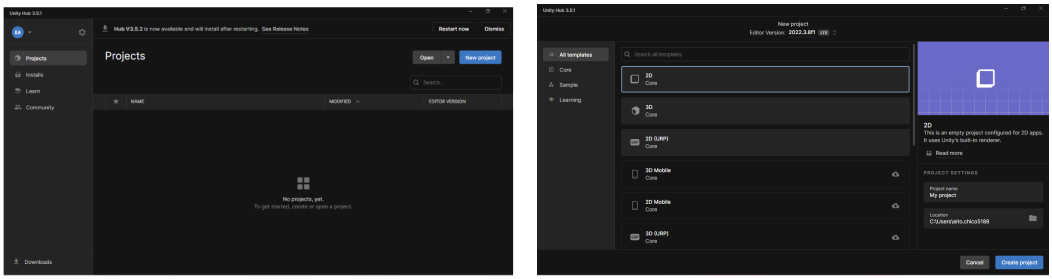
- OS: Windows 7 SP1+, 8, 10, 64-bit versions only; macOS 10.12+; Ubuntu 16.04, 18.04, and CentOS 7.
- GPU: Placa gráfica com recursos do DX10 (Shader Model 4.0).

Obs: No projeto utilizado para criação deste, foi utilizado a Unity versão 2023.3.8f1.

A unity requisita uma interface de programação, algum programa voltado para programação. Um dos mais famosos ,e que foi escolhido para este projeto é o Visual Studio Code. Para fazer seu download, acesse o site oficial da plataforma [Microsoft 2023].

Outra ferramenta necessária para o desenvolvimento será o Chat GPT, que como citado anteriormente, é um chatbot disponível online com capacidade para criar narrativas, orientar no uso de ferramentas (inclusive da unity) e gerar código em c#. Estas características fazem com que o chat GPT seja uma importante ferramenta auxiliar no desenvolvimento de jogos com a unity, especialmente para desenvolvedores iniciantes. Para acessar o Chat GPT, faça login na plataforma online, acessada pelo site oficial [OpenAI 2023].

Agora, para finalziar esta etapa preparatória, é necessária a criação do projeto do jogo na unity. para cria-lo, acesse o unity hub, localize o botão *New Project*, na aba de projetos, vista na figura 3.4(a). Na aba New Project (figura 3.4(b), selecione a opção *2D Core*, renomeie-o como preferir e clique no botão *Create Project*, assim, será criado seu projeto na unity.



(a) Aba Projects na Unity (b) Criação de um novo Projeto

Figura 3.4. Abas Project e New Project na Unity.

Ao fim deste processo de criação, será apresentada a interface de desenvolvimento básica da unity, vista na figura 3.5. Logo mais, a mesma será explicada.

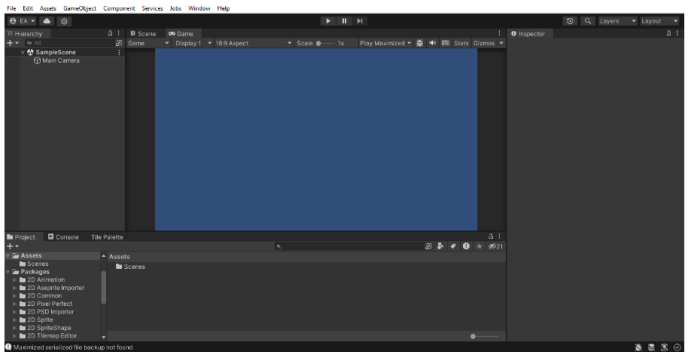


Figura 3.5. InterFace Básica de Desenvolvimento

3.5.2. Interface básica da Unity.

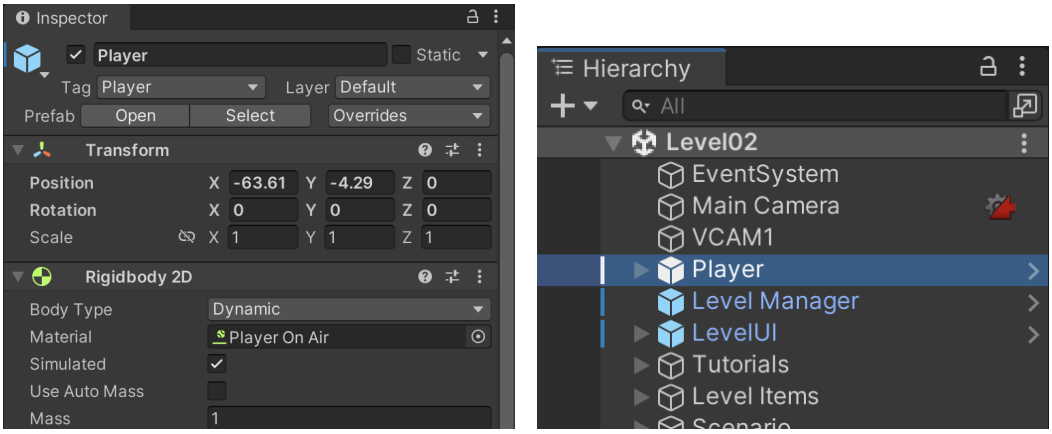
Já com as plataformas prontas, é indispensável a apresentação da interface básica de desenvolvimento da Unity.

3.5.2.1. Inspector

Geralmente localizada no lado direito do monitor, a janela inspector aborda as características e os atributos sobre os objetos. É nela onde são apresentadas as opções de configuração de um objeto dentro da Unity. Como pode-se ver na figura 3.6(a), o objeto “Main Camera”, possui os componentes “Transform”, “Camera” e “Audio Listener”.

3.5.2.2. Hierarchy

Nesta janela, localizada no lado esquerdo do visor, são organizados os objetos presentes na “cena”, tudo aquilo pertencente ao jogo, como players, objetos, paredes, etc. Os objetos podem possuir sub-objetos, que são organizados e separados nesta janela, como é visto na figura 3.6(b).

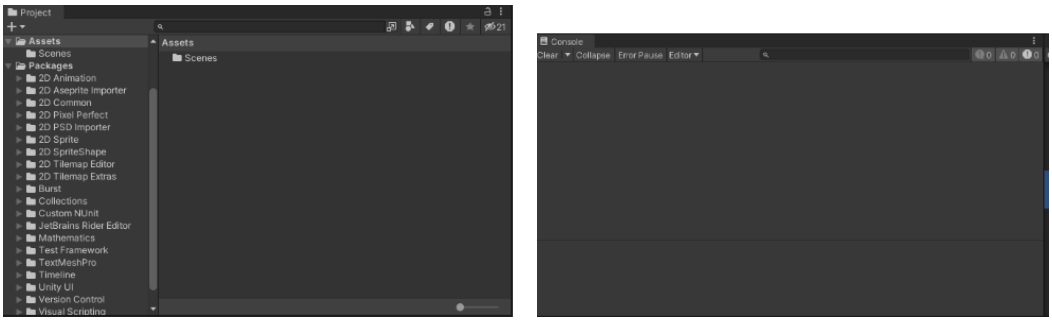


(a) Inspector (b) Hierarchy

Figura 3.6. Janelas Inspector e Hierarchy da Unity.

3.5.2.3. Project

Nesta janela, estão localizados os arquivos do jogo, sendo possível acessar/escolher arquivos. Ao lado pode-se ver na figura 3.7(a) que os arquivos ficam organizados em pastas(ficheiros), tornando mais fácil o acesso



(a) Project (b) Console

Figura 3.7. Janelas Project e Console da Unity.

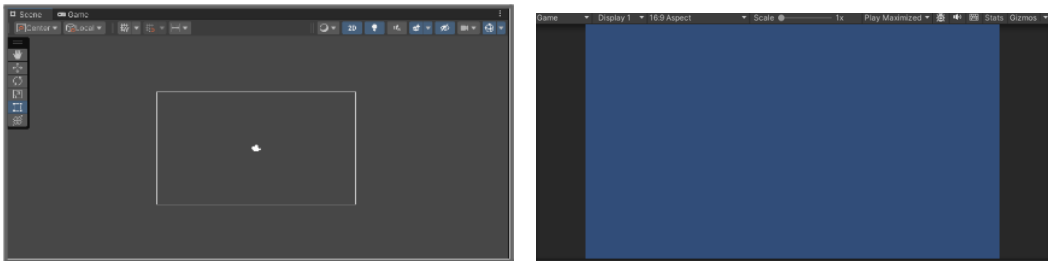
3.5.2.4. Console

A janela console (Figura 3.7(b)), geralmente localizada no canto inferior esquerdo da interface, tem a função de mostrar ao usuário os erros existentes em seu projeto. Caso algum script tenha uma sintaxe errada, algum erro de ortografia, por exemplo, nesta janela terá uma mensagem.

3.5.2.5. Scene

Em todo projeto, é necessário um local para organizar, posicionar, e visualizar objetos das cenas de um jogo. Na unity, esta janela é chamada de Scene (figura 3.8(a)). Nela,

podemos alterar características dos objetos, como suas posições, escalas, dentre outras funções



(a) Scene (b) Game

Figura 3.8. Janelas Scene e Game da Unity.

3.5.2.6. Game

A janela Game possui a função de simular como o jogo ficará,, a partir do que já estiver produzido (figura 3.8(b)).

3.5.3. Tutorial: criando seu primeiro jogo de plataforma 2d com unity e chat GPT

Neste tutorial, será criado um jogo de plataforma 2d que consiste em um player com capacidade de movimentar-se para frente e para trás e pular. O Objetivo do jogo é alcançar o final da fase. Neste minigame serão trabalhados 3 objetos, o chão, o player e o destino.

Na unity, o jogo é representado por uma cena contendo objetos. Estes objetos contêm componentes que representam características do objeto como: posicionamento, detector de colisões, corpo físico, elemento gráfico, elemento sonoro, etc. Além disso, é possível adicionar scripts para acessar estas propriedades e controlar os objetos na cena.

Para criá-los, com o projeto aberto, acesse o menu de contexto (geralmente acessado clicando no botão direito do mouse) na janela de “Hierarchy”, e procurar o menu *create 2D Objects*, em seguida, selecione o sprite desejado, como é visto na figura 3.9. (Utilizado no projeto: circle):

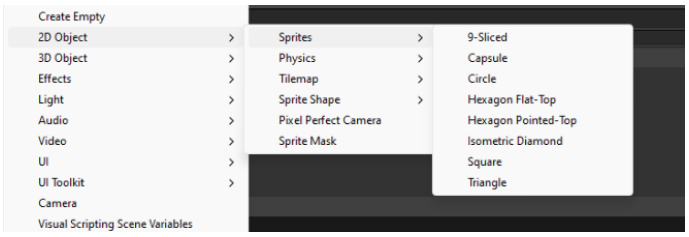


Figura 3.9. Criação do sprite Circle para representar o Player

Com ele criado, percebe-se que nas janelas Hierarchy e Scene foi adicionado um novo objeto, como o nome de Circle. Na Hierarchy, abrindo o menu de contexto, tendo selecionado este objeto, é possível renomeá-lo (recomendável o nome “Player”). Na janela de Inspector (com o objeto do player selecionado) adicione componentes a ele, para isto,

clique no botão “Add Component”, e procure o atributo “RigidyBody2D”, para que o objeto tenha a característica de corpo rígido, com física e etc. Também adicione um atributo “Collider2D”, que possui a função de indicar que o objeto estabelece colisão com outros objetos, tendo as opções de BoxCollider2D e CapsuleCollider2D (neste caso, é utilizado o capsule collider). Com isto, a parte física do player está feita Agora, é preciso que crie o objeto das plataformas, seguindo a mesma lógica. Crie o objeto de “Square” em Hierarchy, adicione RigidyBody2D e desta vez, BoxCollider2D. Assim também o objetivo deve ser criado, utilizando o sprite de “triangle”. Lembre-se de renomeá-lo No objeto da plataforma, acesse o componente RigidyBody2D, e na opção “Body Type”, troque para Static, assim, quando a fase for iniciada, o objeto ficará estático e não sofrerá força gravitacional. Com os objetos criados, pode-se posicionar da forma que for desejado. Também é possível duplicar a plataforma já com seus atributos, acessando o menu de contexto e duplicando. Para a parte lógica do player, inicialmente deve-se criar um script de controle do player. Crie um folder(menu de contexto, “Create”, “Folder”) na janela Project, com o nome de “script”, nela, acesse o menu de contexto, e selecione “Create”, “C# Script”, nomeie-o de PlayerController, pois ele possuirá o algoritmo de locomoção horizontal do player. É necessário associar o script ao objeto do player (para isto, vá na janela inspector do objeto, procure a opção “Add component” e procure o script desejado). Agora, o chat GPT será utilizado para criação de um simples prompt de movimentação horizontal do player. Neste caso, foi utilizado o seguinte prompt: "Olá, preciso de um simples prompt de movimentação horizontal de um player em um jogo de plataforma 2D na unity utilizando C#".

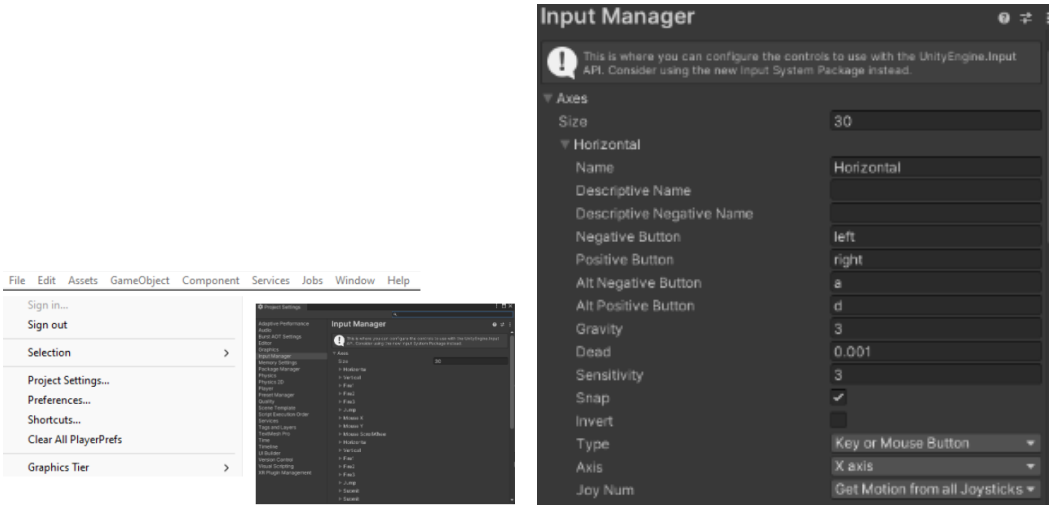
O mesmo respondeu anexando o algoritmo 3.10

Algoritmo 3.10. Movimentação Horizontal

```
using UnityEngine;

public class PlayerController : MonoBehaviour{
    public float speed = 5.0f;
    private Rigidbody2D rb;

    private void Start(){
        rb = GetComponent<Rigidbody2D>();
    }

    private void Update(){
        float moveInput = Input.GetAxis("Horizontal");
        Vector2 moveVelocity = new Vector2(moveInput *
           speed, rb.velocity.y);
        rb.velocity = moveVelocity;
    }
}
```

Antes de tudo é preciso entender que quando criado um projeto, a Unity gera uma série de funcionalidades. Uma delas é a pré-alocação de funções para os botões no teclado. Isto funciona da seguinte forma: Na unity, na barra superior, vá no botão de

“Edit” e procure o menu “Project Settings”. Clicando nele, será aberta uma nova janela com o mesmo nome, com diversas informações, nela, procure no menu esquerdo a opção “Input Manager”, como visto na figura 3.10(a).



(a) InputManager (b) Inputs Horizontais

Figura 3.10. Aba Input Manager e Inputs Horizontais

Neste menu, são apresentadas as configurações de entrada. Em cada um dos tópicos, temos uma pré-configuração de botões. Em horizontal, por exemplo, temos os botões que são chamados para movimentos horizontais, como é observado na figura 3.10(b). Em Positive button, temos a tecla “d” configurada, e em Negative button temos a tecla “a”. Isto significa que quando apertada (dependendo do código utilizado), a tecla “d”, o player andará para frente, e quando apertada “a”, andará para trás.

Com o Visual Studio Code aberto no script desejado, copie do chatgpt o código disponibilizado e cole-o no script. Com o texto pronto, sempre é necessário salvá-lo. Esa função pode ser feita pelo atalho no teclado (Ctrl + S).

Agora, é necessário um prompt para o player, com a função de pular entre as plataformas. Para criá-lo, faça como o anterior, nomeie-o de PlayerJump e associe-o ao player. Neste caso, foi utilizado o seguinte prompt: "Crie agora um script de pulo para o player, utilizando o input "Jump". O código resultante da requisição para o chat GPT é apresentado no Algoritmo 3.11.

Algoritmo 3.11. Atributos do script PlayerJump

```
1 public float jumpForce = 8.0f;
2 private Rigidbody2D rb;
3
4 private void Update(){
5     if (Input.GetButtonDown("Jump"))
6     {
7         rb.AddForce(Vector2.up * jumpForce, ForceMode2D.
              Impulse);
8     }
```

```
}
```

No Algoritmo 3.11, são criados dois atributos: jumpForce e rb. O jumpForce corresponde à força que irá impulsionar o player para cima. Já o rb representa o corpo físico do objeto onde será aplicada a força.

Neste passo, faça como no script anterior, copie, cole, e depois salve-o. Neste ponto, nosso player já se movimenta horizontalmente e já consegue pular. Para testar isto, a Unity disponibiliza a janela game onde é possível executar um *preview* do jogo.

Como em todo jogo, este também necessita de um aviso que o player alcançou seu objetivo. Na unity, isto pode ser feito por meio dos objetos chamados de panels. Para criá-lo, pode-se pedir um passo a passo ao Chat GPT, é propiciado um breve tutorial de como produzi-lo. Com o prompt: "Tenho um objeto que será o objetivo final do player, preciso que apareça na tela do jogador, um painel dizendo que ele ganhou ou jogo quando quando o mesmo colidir com o objetivo, como posso fazer isto?". A IA respondeu anexando as seguintes estruturas de algoritmo:

Algoritmo 3.12. Variáveis do algoritmo de Level End

```
public GameObject victoryPanel;
private bool hasWon = false;
```

No algoritmo 3.12, vemos a parte de variáveis e atributos do script de level end

Algoritmo 3.13. Métodos de Colisão

```
private void OnTriggerEnter2D(Collider2D other){
        if (other.CompareTag("Player") && !hasWon){
            hasWon = true;
            ShowVictoryPanel();
        }
    }
```

Algoritmo 3.14. Algoritmo Level End

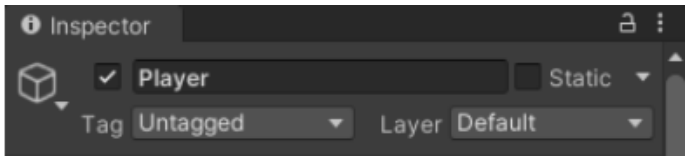
```

 private void ShowVictoryPanel(){
        victoryPanel.SetActive(true);
    }
```

No algoritmo 3.14, observamos o método "ShowVictoryPanel", que ativa o painel de vitória.

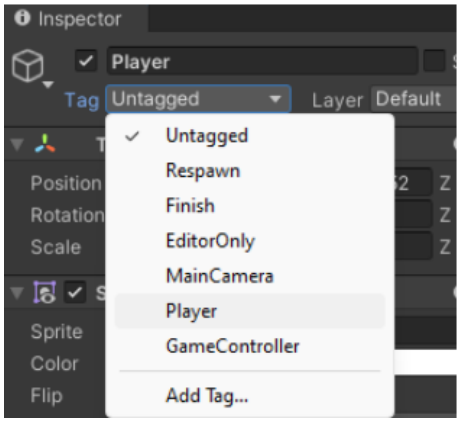
Com estas instruções, resta ainda a criaç ao de alguns componentes. Primeiramente, você precisa criar a UI (User Interface) geral do jogo. Para isto, crie um objeto do tipo canva, visto no menu de criação "UI".

Renomeie-o para UI, e logo após isto, com ele selecionado, crie um novo objeto do tipo *panel*, um pouco acima de canvas. Este painel criado se torna “herdeiro” do canva. No inspector, deve-se desmarcar a caixa ao lado do nome do objeto, assim, o objeto inicia

a fase estando desativado, e só é ativado caso algo seja estabelecido. Agora, com o script "GameManager" ativado, faz-se como nos outros scripts anteriores, todavia, associe este ao objeto de objetivo. Para que este script funcione, é necessário associar a tag player ao objeto player. No inspector do player, selecione a opção "Tag", e nela, escolha a tag player, como é visto nas figuras 3.11(a) e 3.11(b). Isto serve para diferenciar o player de objetos comuns.



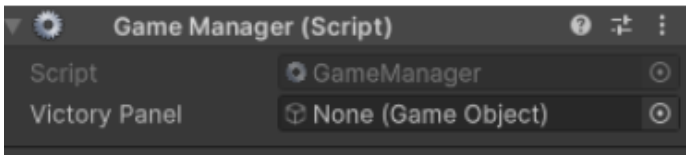
(a) Untagged



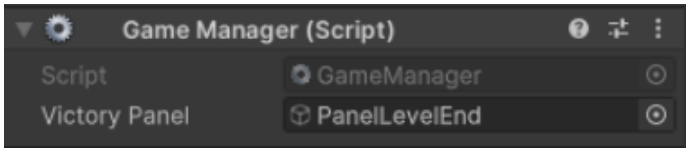
(b) Tag Player

Figura 3.11. Player com a tag "Player"

Além disso, também é necessário atribuir o panel ao gameobject do script. No inspector do objetivo com o script GameManeger, arraste o panel para a aba do componente, nas figuras 3.12(a) e 3.12(b). Verifique se no componente de capsule collider do objetivo, a caixa "Is trigger" está marcada(figura 3.13).



(a) Seleção de Panel Vazia



(b) Painel com PanelLevelEnd selecionado

Figura 3.12. Painel de Seleção de Objeto

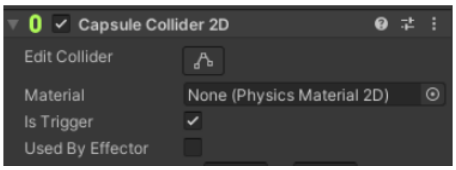
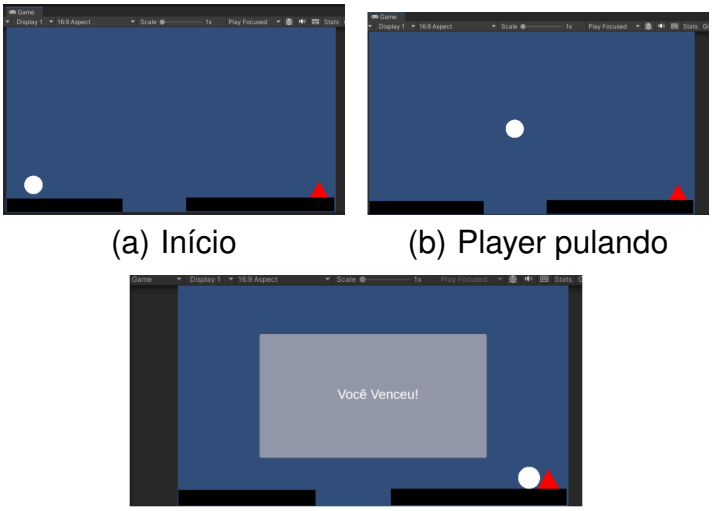


Figura 3.13. Opção IsTrigger



(a) Início (b) Player pulando

(c) Final

Figura 3.14. Capturas de tela do protótipo de jogo de plataforma funcional.

Por fim, é necessário a personalização do painel, como adição de texto, sendo feita por meio do componente adicionável *TextMeshPRO*. Adicione o componente no painel, como feito com os outros componentes adicionados. Na Hierarchy, com o painel selecionado, adicione o componente "TextMeshPRO", localizado no menu *UI*. Para personalizalo, vá ao inspetor do TextMeshPRO adicionado. A Figura 3.14 apresenta as capturas de tela de como o jogo deverá se comportar após a realização dos passsos descritos anteriormente.

3.6. Conclusão

Neste capítulo, foram explorados elementos essenciais que compõem um jogo, como as mecânicas, regras e desafios que o tornam envolvente. Tamém Investigamos as ferramentas e motores de jogo populares usados C#. Ao criar um estudo de caso para um jogo de plataforma, os participantes tiveram a oportunidade de aplicar o conhecimento adquirido em um projeto prático.

Portanto, espera-se que os conhecimentos adquiridos ao longo deste minicurso sejam não apenas valiosos para a compreensão do desenvolvimento de jogos, mas também funcionem como uma porta de entrada para aqueles que desejam ingressar nessa área profissionalmente.

Referências

[OpenAI 2023] OpenAI (2023). Chatgpt. Disponível em: https://chat.openai.com. Acesso em: 04 de outubro 2023.

[Arts 2000] Arts, E. (2000). The sims. Plataformas: Microsoft Windows, MacOS, Linux, PlayStation 2, Nintendo GameCube, Nintendo DS, Xbox, Mac OS Classic.

[Bethesda 20] Bethesda (20). The elder scrolls v: Skyrim. Plataformas: Nintendo Switch, PlayStation 4, Xbox One, Steam.

[Entertainment 2010] Entertainment, B. (2010). Starcraft ii. Plataforma: MacOS, Microsoft Windows.

[Epic Games 2023] Epic Games (2023). Unreal engine. Disponível em: https://www.unrealengine.com/pt-BR. Acesso em: 04 de outubro 2023.

[King 2012] King (2012). Candy crush saga. Plataformas: Android, IOS, Microsoft Windows.

[Mark Overmars 2023] Mark Overmars (2023). Gamemaker. Disponível em: https://gamemaker.io/pt-BR. Acesso em: 04 de outubro 2023.

[Michelle Rouhof 2023] Michelle Rouhof (2023). Newzoo. Disponível em: https://newzoo.com/. Acesso em: 21 de setembro 2023.

[Microsoft 2023] Microsoft (2023). Visual studio code. Disponível em: https://code.visualstudio.com. Acesso em: 04 de outubro 2023.

[Mojang 2011] Mojang (2011). Minecraft. Plataformas: IOS, Android, PlayStation 4, Nintendo Switch, Xbox One, Xbox Series X e Series S, Microsoft Windows.

[Nintendo 1983] Nintendo (1983). Super mario bros. Plataformas: Nintendo Switch, Nintendo Entertainment System.

[Nintendo 1986] Nintendo (1986). The legend of zelda. Plataformas: Nintendo Entertainment System.

[Nintendo 1987] Nintendo (1987). Mega man. Plataformas: Android, iOS, Nintendo Entertainment System.

[Pajitnov 1984] Pajitnov, A. (1984). Tetris. Plataformas: Game Boy, Nintendo Entertainment System.

[RED 2015] RED, C. P. (2015). The witcher 3: Wild hunt. Plataformas: PlayStation 5, PlayStation 4, Nintendo Switch, Xbox One, Xbox Series X e Series S, Microsoft Windows.

[SEGA 1991] SEGA (1991). Sonic the hedgehog. Plataformas: Mega Drive, Android, SG-1000 Mark III, PC, Playstation, Xbox.

[Sony 2021] Sony (2021). Suplemento de informações para os resultados financeiros consolidados para o quarto trimestre encerrado em 31 de março de 2021. Technical report, Sony.

[SquareSoft 1992] SquareSoft (1992). Final fantasy v. Plataforma: Game Boy Advance.

[SX group 2023] SX group (2023). Pesquisa game brasil. Disponível em: `https://www.pesquisagamebrasil.com.br`. Acesso em: 30 de setembro 2023.

[Thorson 2018] Thorson, M. (2018). Celeste. Plataformas: Steam, Epic Games, Xbox, PlayStation, Switch.

[Unity Technologies 2023] Unity Technologies (2023). Plataforma de desenvolvimento em tempo real do unity. Disponível em: `https://unity.com/pt`. Acesso em: 04 de outubro 2023.