ATIVIDADE UNITY

MATHEUS HENRIQUE BUTKOSKI SILVA

1) Uma Game Engine é um software que fornece ferramentas e recursos para facilitar o desenvolvimento de jogos eletrônicos. Suas funções principais são lidar com gráficos, física, áudio, inteligência artificial, cenas e outros aspectos técnicos, permitindo que os desenvolvedores foquem na criação do jogo utilizando as ferramentas disponibilizadas. Isso acelera o processo de desenvolvimento e torna possível criar jogos para diversas plataformas.

2) GameObjects são os blocos de construção fundamentais para criar qualquer coisa em Unity, e eles permitem que você crie ambientes interativos e jogos complexos ao unir diferentes componentes e scripts para definir o comportamento e a aparência de objetos no seu jogo.

3) Cada GameObject possui o componente "Transform", esse é essencial para definir a posição, rotação e escala de um objeto em um jogo. Ele controla onde o objeto está localizado, como ele está orientado e quão grande ele é no espaço do jogo. Ele é fundamental para posicionar e dar forma aos objetos em uma cena.

4) O componente "Collider" em Unity é usado para definir a geometria de colisão de um objeto em um jogo. Ele permite que objetos interajam realisticamente detectando colisões ou sobreposições com outros objetos. Isso é essencial para criar física, detecção de colisões e interações entre elementos do jogo.

5) A propriedade "Is Trigger" do componente “Collider” em Unity transforma o Collider em um detector de colisão que não impede objetos de passarem um pelo outro, mas dispara eventos quando ocorre uma sobreposição. É usado para criar interações específicas no jogo, como acionar eventos, cutscenes ou ações quando um objeto entra em uma área delimitada pelo “Collider”, sem bloquear fisicamente o movimento dos objetos. Isso é útil para diversas situações, como áreas de coleta de itens, zonas de ativação de armadilhas e portais de teletransporte.

6) O componente "Rigidbody2D" em Unity é usado para adicionar física realista a objetos 2D em um jogo. Ele permite que esses objetos se movam, colidam e interajam com base nas leis da física. O Rigidbody2D controla o movimento, a gravidade, as colisões e as forças aplicadas aos objetos, tornando possível criar jogos 2D com física realista e interações dinâmicas entre elementos do jogo.

7) O componente "Sprite Renderer" em Unity é usado para renderizar sprites 2D na cena do jogo. Sua função principal é exibir imagens ou texturas 2D, permitindo a personalização da aparência visual dos objetos no ambiente 2D. Ele também controla a ordem de renderização dos sprites e oferece opções para ajustar cores, materiais e outros efeitos visuais.

8) O componente "Animator" em Unity é usado para criar e controlar animações em objetos, personagens e elementos de um jogo. Com o Animator, você pode sincronizar animações com eventos de jogo, criar transições suaves entre estados e gerenciar animações de forma interativa, tornando-o essencial para dar vida e movimento aos elementos em um jogo.

9) Os dois principais métodos de um script em Unity são "Start()" e "Update()". "Start()" é chamado uma vez no início e é usado para configurações iniciais e aquisição de referências. "Update()" é chamado a cada quadro e é usado para lógica em tempo real, como movimento, entrada do jogador e verificações contínuas. O "Start()" é para configuração inicial, e o "Update()" é para lógica em andamento durante o jogo.

10) A função Time.deltaTime em Unity é usada para medir o tempo decorrido entre quadros consecutivos do jogo. Seu objetivo é permitir que o código seja executado com base no tempo, tornando os movimentos e comportamentos independentes da taxa de quadros (FPS) do jogo. Isso garante que as ações sejam consistentes em diferentes sistemas, proporcionando uma experiência de jogo suave e equilibrada. Geralmente, é multiplicada pela velocidade, assim permitindo, por exemplo, que o código mova um objeto para frente ou para trás.

11) Os "Inputs" em Unity referem-se à coleta de dados de entrada do jogador, como pressionamentos de teclas, cliques do mouse e toques na tela. Sua função principal é permitir que o jogo responda às ações do jogador e interaja com ele. Isso inclui controlar personagens, ativar ações, gerenciar a câmera, interagir com a interface do usuário e desencadear eventos no jogo. Os Inputs são essenciais para criar uma experiência de jogo envolvente e dinâmica, tornando a interação entre o jogador e o jogo possível e satisfatória.

12) A diferença principal entre Input.GetAxisRaw e Input.GetAxis em Unity está na suavização dos valores de entrada e no intervalo de valores retornados. GetAxisRaw fornece valores brutos sem suavização, retornando -1, 0 ou 1, adequado para controles digitais. Por outro lado, GetAxis aplica suavização, permitindo valores entre -1 e 1, incluindo frações, útil para controles analógicos. A escolha entre eles depende do tipo de controle e do comportamento desejado no jogo.

13) Utilizando as funções getAxisRaw ou getAxis, se tivermos um número negativo, o objeto estará se movendo para a esquerda. Por outro lado, se tivermos um número positivo, o objeto estará se movendo para a direita, caso o valor seja 0, o objeto estará parado.

14) O método Update() em Unity é usado para conter a lógica do jogo que precisa ser atualizada a cada quadro do jogo. Ele é executado a cada quadro (geralmente 60 vezes por segundo) e é usado para processar entrada do jogador, atualizar a posição de objetos, verificar condições de jogo, manter atualizações contínuas e executar lógica em tempo real.

15) Devemos utilizar a propriedade FlipX ou FlipY do componente Sprite Renderer, dessa forma será possível inverter os eixos de um objeto.

16)
```
void move(){
    Vector3 moviment = new Vector3(Input.GetAxisRaw("Horizontal"), 0, 0);
    transform.position += moviment * speed * Time.deltaTime;
    }
```

17)
```
void jump(){
  skin.GetComponent<Animator>().Play("Jump",-1);
   rigid.AddForce(new Vector2(0, impulse), ForceMode2D.Impulse);
}
```

18) O método “OnCollisionEnter2D” em Unity é um callback que é chamado automaticamente quando ocorre uma colisão 2D entre o objeto ao qual o script está anexado e outro objeto 2D. Seu objetivo principal é permitir que você responda a colisões 2D e execute ações específicas quando elas ocorrem. Isso é útil para detectar colisões, ativar eventos, alterar comportamentos e criar efeitos visuais em resposta a colisões no jogo.
19) O método “OnCollisionEnter2D” é usado para detectar e responder à primeira colisão entre objetos 2D, enquanto “OnCollisionStay2D” é usado para lidar com ações contínuas e comportamentos que ocorrem enquanto a colisão persiste. A escolha entre esses métodos depende das necessidades específicas de detecção e resposta de colisões no jogo.

20) O método “CompareTag()” em Unity é usado para verificar se um objeto tem uma tag específica. Ele é frequentemente utilizado em eventos de colisão para identificar objetos com os quais você deseja interagir. A função principal é permitir que você tome decisões ou realize ações com base na comparação de tags, facilitando a identificação de objetos específicos no jogo.