Unity

Movimentação do personagem

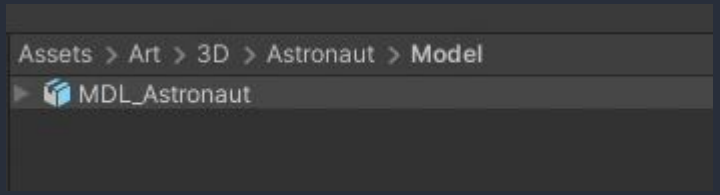
Mecanim

Neste módulo vamos revisar o funcionamento do Mecanim.

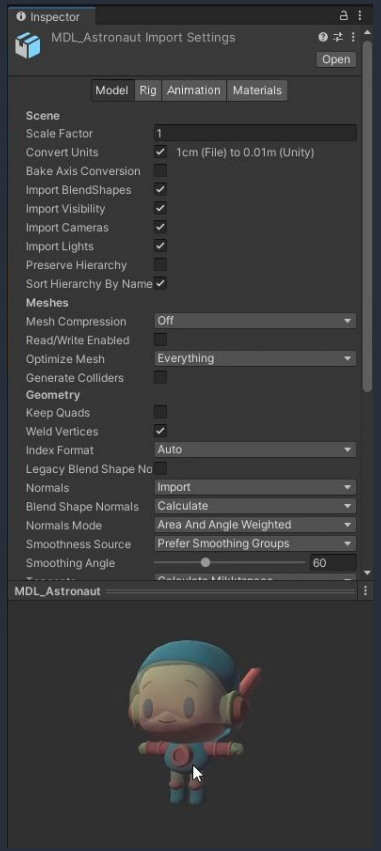
O Mecanim é o sistema de animação poderoso e flexível da Unity, projetado para simplificar a criação de animações complexas para personagens e objetos em jogos. Ele oferece uma interface visual intuitiva para criar transições suaves entre diferentes estados de animação, como correr, pular e atacar, facilitando a implementação de comportamentos de personagens realistas. Além disso, o Mecanim suporta a blendagem de animações, a criação de árvores de estados e a integração de animações 2D e 3D, tornando-o uma ferramenta essencial para desenvolvedores de jogos que buscam criar experiências de jogo imersivas e dinâmicas na Unity.

Mecanim

Acesse a pasta abaixo do projeto:

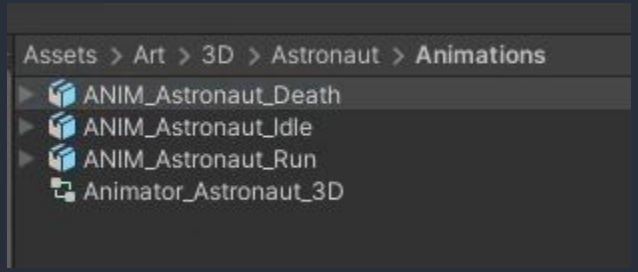


Nela é possível observar o personagem principal.



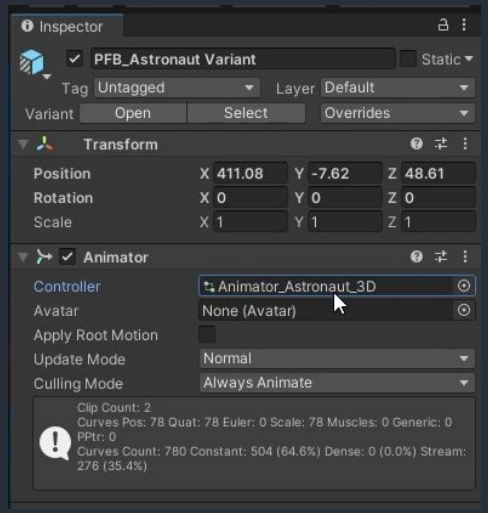
Mecanim

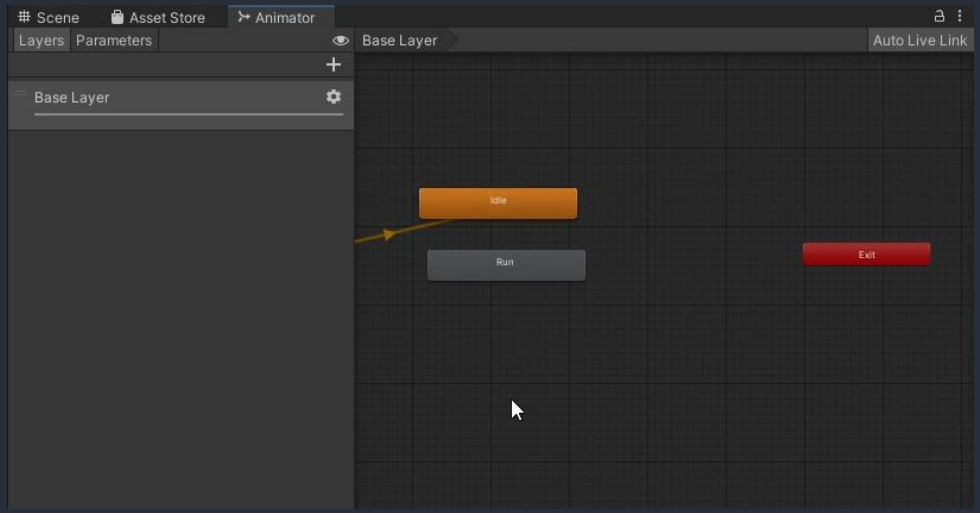
E na pasta abaixo temos todas as animações do nosso Astronauta



Mecanim

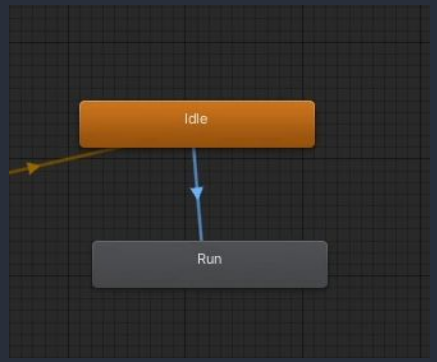
O component Animator é o controlador da máquina de estados da animação. É através dele que vamos poder observar e configurar os diferentes estados das animações do nosso herói.

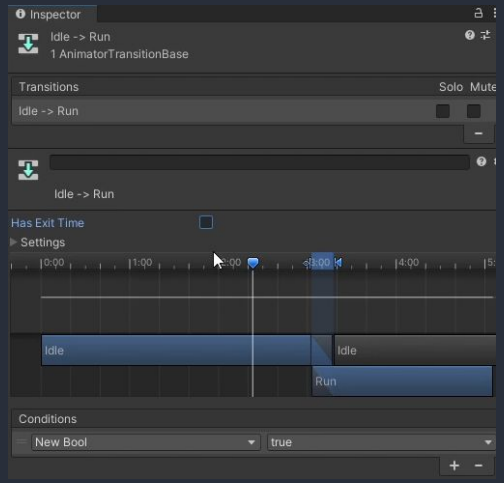




Mecanim

Para fazer transições entre as animações, clique com o botão direito na animação e clique em “Make Transition”. Depois, selecione a animação seguinte que quer fazer a transição.





Implementação da movimentação

Agora vamos começar implementar a movimentação. Veja esse vídeo do jogo Super Lucky's Tale.
Vamos analisar a movimentação do personagem para aplicarmos no nosso projeto.

Implementação da movimentação

Crie um script chamado Player. Adicione nele o seguinte código:

```
public CharacterController characterController;
public float speed = 1f;
public float turnSpeed = 1f;
public float gravity = 9.8f;

private float vSpeed = 0f;

void Update()
{
    transform.Rotate(0, Input.GetAxis("Horizontal") * turnSpeed * Time.deltaTime, 0);

    var inputAxisVertical = Input.GetAxis("Vertical");
    var speedVector = transform.forward * inputAxisVertical * speed;

    vSpeed  -= gravity * Time.deltaTime;
    speedVector.y = vSpeed;

    characterController.Move(speedVector * Time.deltaTime);
}
```

Implementação da movimentação

Crie um GameObject na cena, e de o nome de Player. Dentro dele, crie um cubo, que servirá como representação visual do nosso player por enquanto.

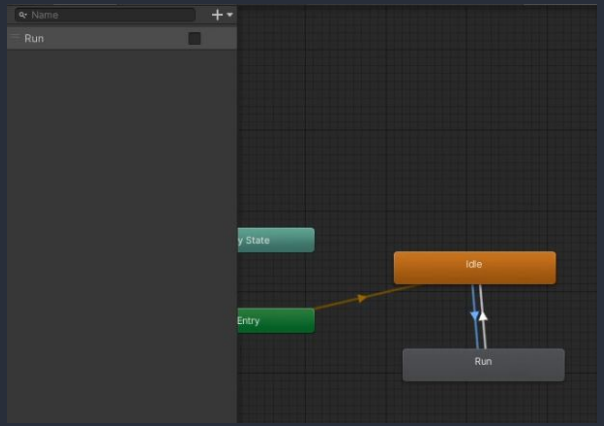
Implementação da movimentação

Adicione também no mesmo GameObject, a configuração abaixo:

Implementação da movimentação

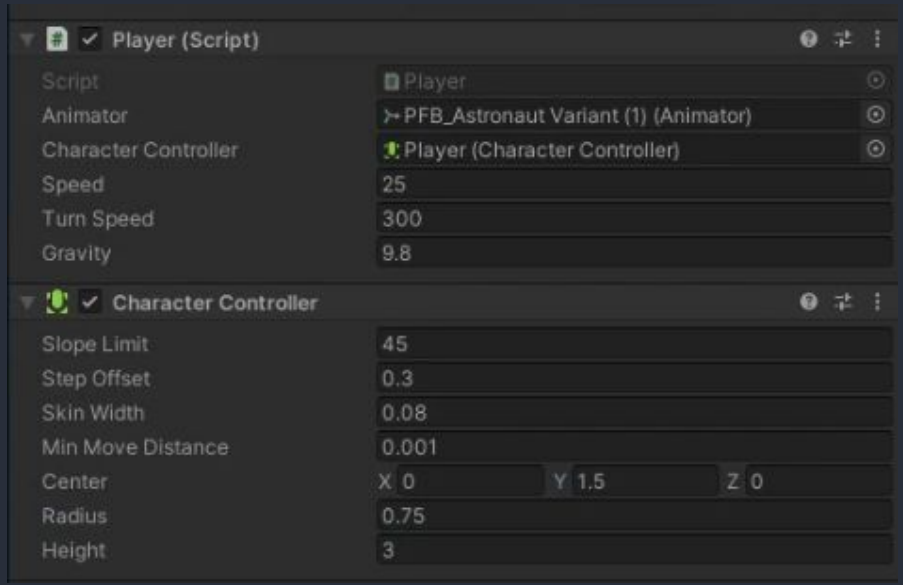
Agora vamos colocar o nosso personagem no jogo. Selecione o prefab do Astronauta, e coloque-o dentro do GameObject Player.

Dentro do animator do astronauta, crie um setup como o da imagem abaixo: 2 transições de Idle <-> Run e um parâmetro “Run” para controlar as transições.



Implementação da movimentação

Atualize as configurações como os valores abaixo:



Implementação da movimentação

No script do Player, adicione o seguinte:

```
public Animator animator;

Dentro do método update, adicionar:
void Update()
{
   if(inputAxisVertical !=0)
   {
        animator.SetBool(“Run”, true);
   }
   else
   {
        animator.SetBool(“Run”, false);
    }
}
```

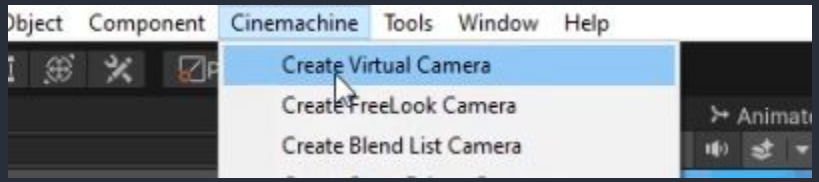
Cinemachine

Agora que nosso jogador já consegue se movimentar pelo cenário, vamos adicionar um controlador de câmera: a Cinemachine.

A Cinemachine é uma extensão da Unity que simplifica o controle e criação de câmeras em jogos, permitindo efeitos cinematográficos e transições suaves. Ela facilita a configuração de câmeras, tornando as experiências de jogo mais envolventes e visuais impressionantes.

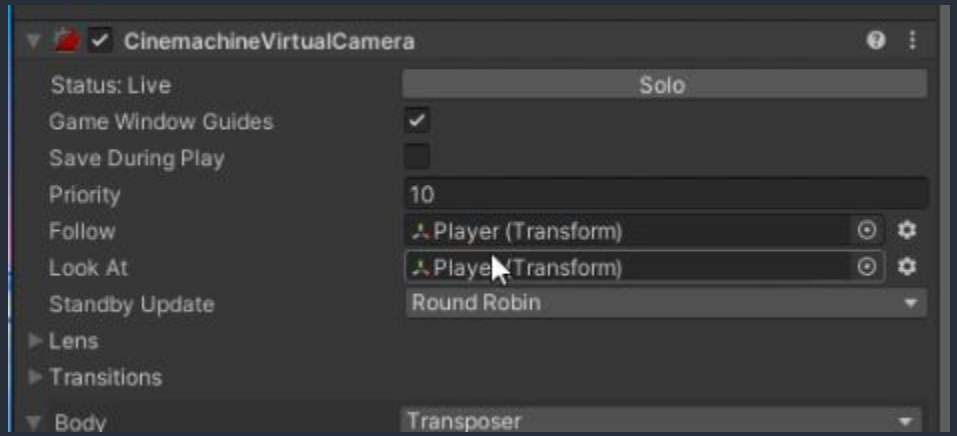
Cinemachine

Para adicionar a cinemachine que queremos ao projeto, selecione como na imagem abaixo:



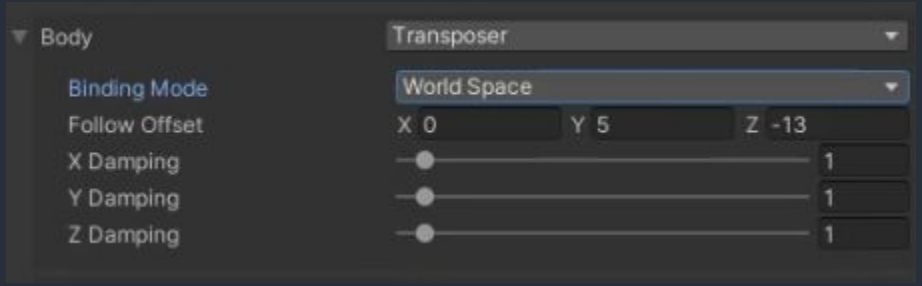
Cinemachine

Atualize as referências dos itens Follow e Look At com o gameobject do Player



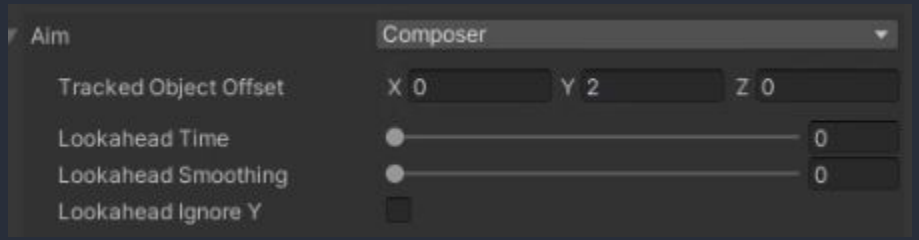
Cinemachine

Atualize a sessão **Body** com a configuração abaixo. Isso ajustará a posição da câmera para o player.



Cinemachine

Também vamos ajustar a sessão “Aim”:



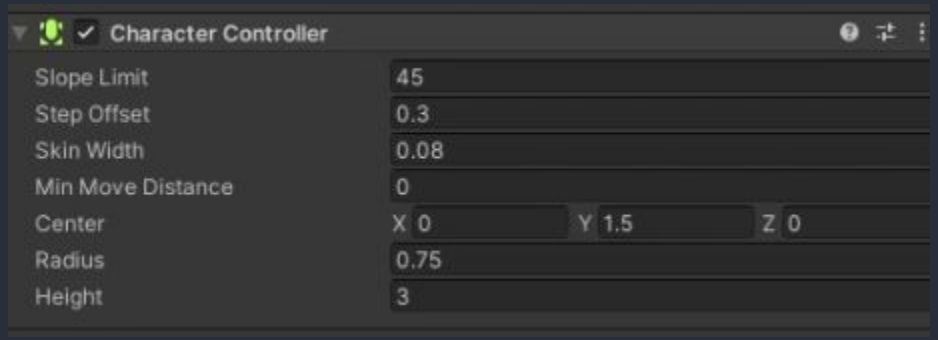
Pulo

Selecionar o script Player e adicionar o Pulo ao nosso personagem.

```
public float jumpSpeed = 15f;

void Update()
{

    var speedVector = transform.forward * inputAxisVertical * speed;

   if(characterController.isGrounded)
    {
        vSpeed = 0;
        if(Input.GetKeyDown(KeyCode.Space))
        {
            vSpeed = jumpSpeed;
        }

    }

     vSpeed = gravity * Time.deltaTime;
}
```

Pulo

No CharacterController, coloque o MinMoveDistance para 0:



Corrida

Para fazer nosso personagem correr, vamos adicionar o seguinte código:

```
[Header("Run Setup")]
 public KeyCode keyRun = KeyCode.LeftShift;
 public float speedRun = 1.5f;

 void Update()
 {
     var isWalking = inputAxisVertical != 0;
     if(isWalking)
     {
         if(Input.GetKey(keyRun))
         {
             speedVector *= speedRun;
             animator.speed = speedRun;
         }
         else
         {
             animator.speed = 1;
         }
     }
 }
```