

# Mòdul 8

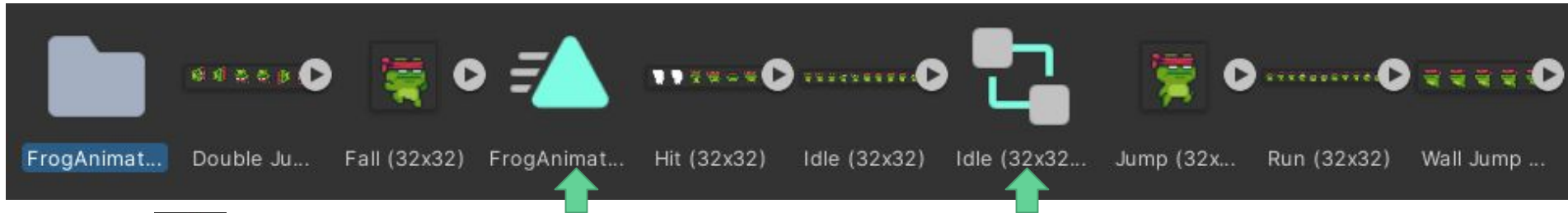
---



UF 3: desenvolupament de jocs per a dispositius mòbils. 30 hores

# Animació del personatge

# Animació del personatge

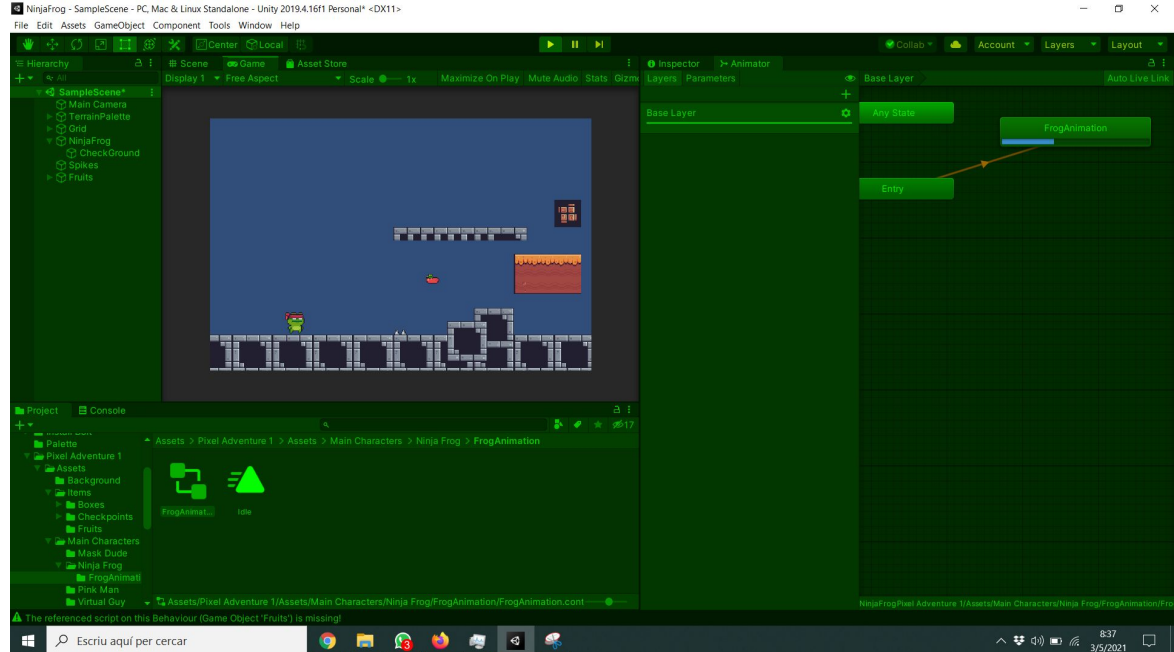
- A la carpeta Assets crearem una nova carpeta que es dirà Animations → MainCharacter. Allà hi afegirem totes les animacions del personatge principal.
- Si arrosseguem els frames de l'animació a l'escena i guardem l'animació a la carpeta se'ns crearan dos arxius




-  Aquest arxiu l'anomenarem Idle. Serà l'animació per quan estigui parat.
-  Aquest arxiu l'anomenarem FrogAnimation. Serà el controlador d'animacions en cada moment

# Animació del personatge

- Si obrim l'arxiu d'animació i fem click al play veurem el que l'animació de IDLE es carrega de manera constant. El que voldrem aconseguir és modificar l'animació en funció de l'estat del personatge.



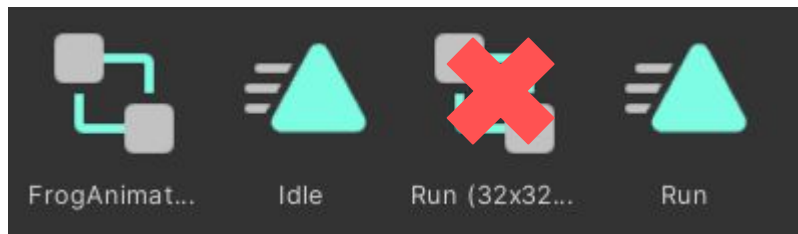
# Animació del personatge. RIGHT & LEFT

- Si fem click al personatge veurem que té un component que es diu SpriteRenderer. Aquest té una propietat que es diu FLIP, si seleccionem la X veurem que l'animació es gira a l'esquerra. 
- Obrirem el script del moviment i declararem el component SpriteRender tal i com fem amb el Rigidbody.
- Farem que quan vagi a l'esquerra la propietat flipX = true i quan vagi a la dreta flipX = false.

```
spriteRenderer.flipX = false;
```

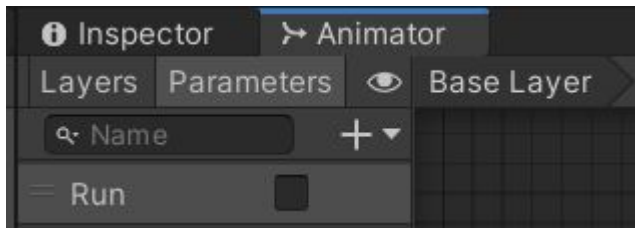
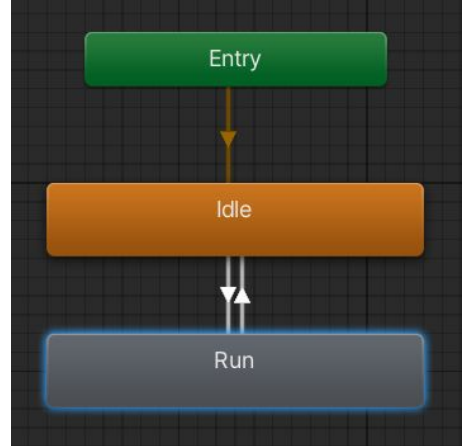
# Animació del personatge. RUN

- Arrosseguem a escena l'animació de run i guardem dins la carpeta d'animacions creada anteriorment. Anomenem l'animació amb un nom identificatiu.
- Eliminem el personatge de l'escena
- A la carpeta d'animacions eliminem el controlador de les animacions de manera que ens ha de quedar així la carpeta:



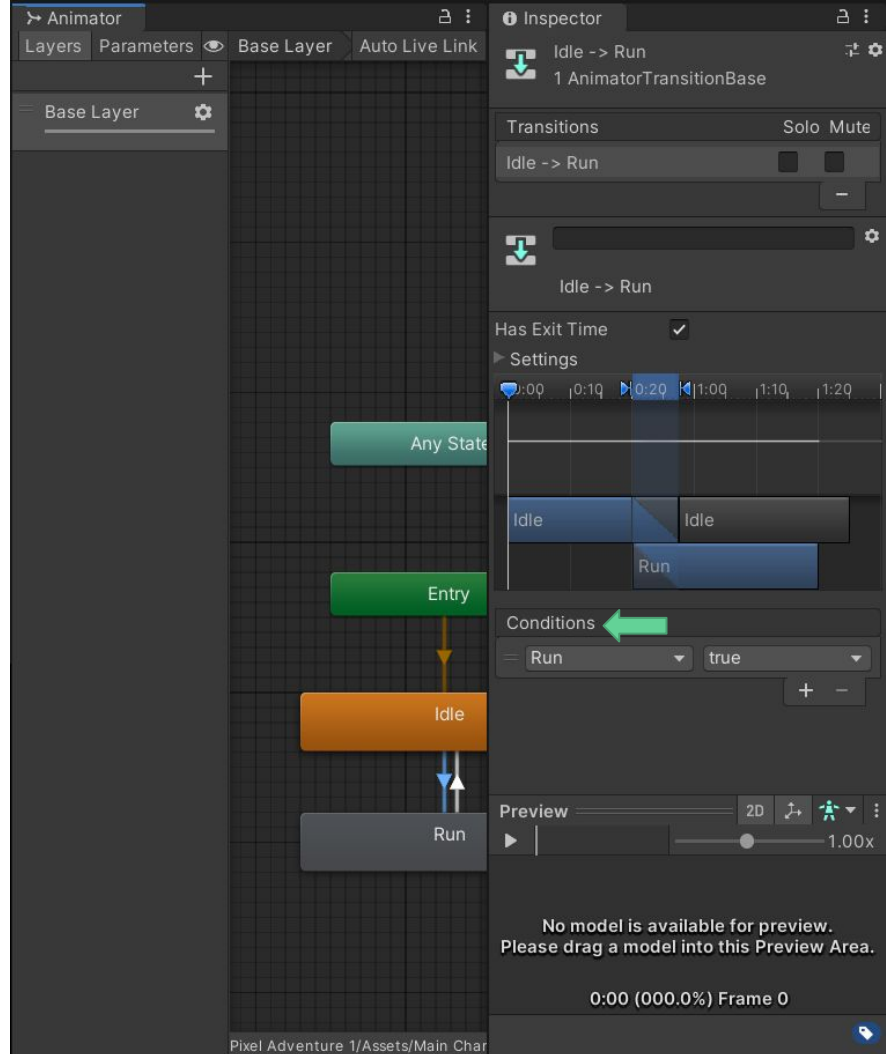
# Animació del personatge. RUN

- A l'arbre d'animació hi arrossegarem l'animació creada.
- Voldrem que quan estigui parat estigui fent l'animació IDLE però que quan comenci a córrer passi a l'animació RUN.
- Hem de fer dues transicions, fent click dret a Idle → Make Transition i el mateix des de run.
- Crearem un paràmetre booleà que ens indiqui quan està corrent i quan no, d'aquesta manera podrem fer la transició.



# Animació del personatge. RUN

- Hem determinar la condició de la transició, per a fer-ho fem click a la fletxeta i anem a la pestanya Inspector i li assignem la condició Run a true o false en funció de l'estat.
- Fes-ho pels dos estats de l'animació.





# Animació del personatge. RUN

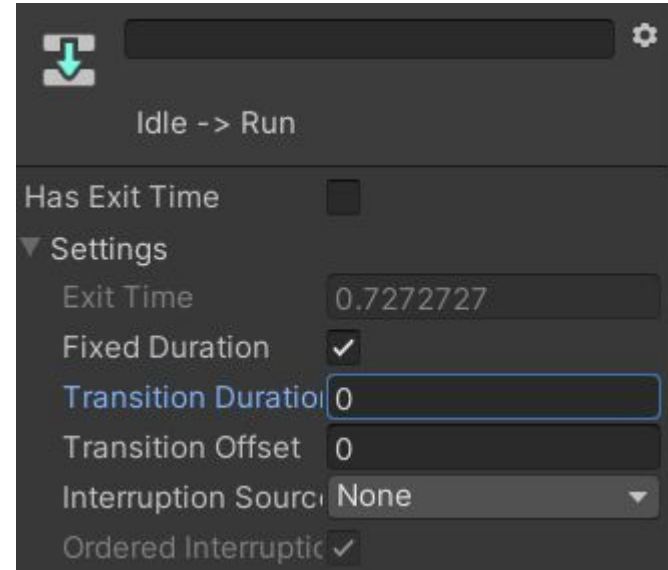
- Ara només ens queda dir-li mitjançant l'script que quan corri es modifiqui la variable boleana que hem creat.
- Per a fer-ho anem al script PlayerMove i declarem el component Animation tal i com hem fet amb el Rigidbody i amb el SpriteRenderer.
- Quan es mogui a dreta i a esquerra voldrem que la variable Run estigui a true, per contra, quan no s'estigui movent, voldrem que estigui a false.

```
animator.SetBool("Run", true);
```

- Fixeu-vos que l'animació és estranya i que comença a córrer una mica més tard del que nosaltres ho indiquem. Això és perquè realitza tota l'animació IDLE abans de començar a córrer.

# Animació del personatge. RUN

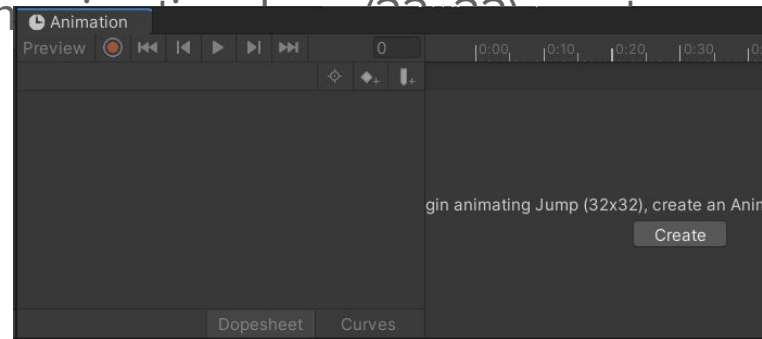
- Farem click a la fletxeta de transició, desmarcarem l'opció Has Exit Time i posarem a 0 la Transition Duration aquí pots posar un valor baix si vols que hi hagi una mica de transició entre una animació i una altra.
- Fes-ho per a les dues transicions.



# Animació del personatge. JUMP

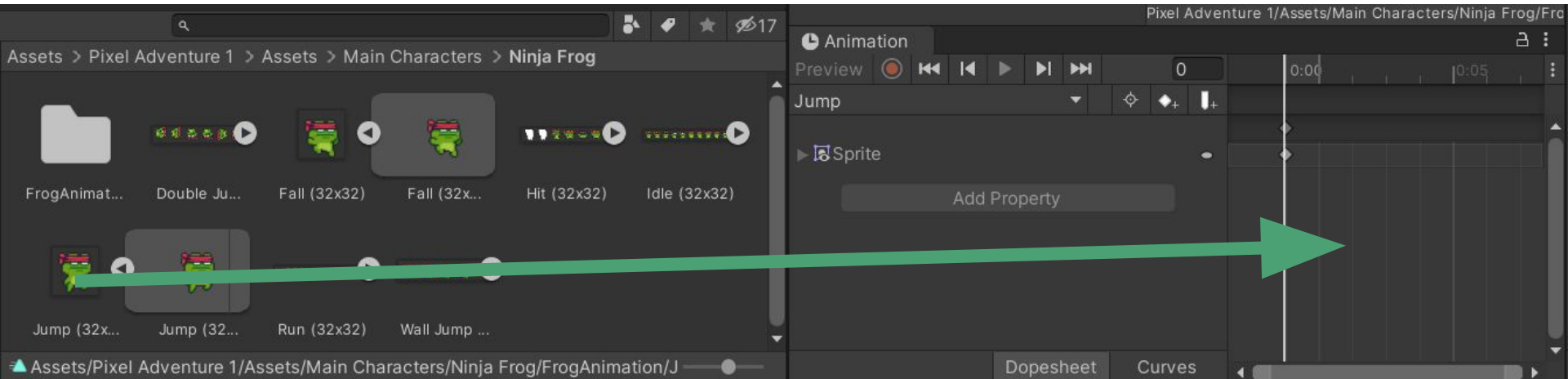
Per crear l'animació del salt farem el mateix que hem fet per córrer, tenint en compte una única cosa. Els assets no ens ofereixen una animació ja feta del salt, és per això que l'hauréu de crear mitjançant l'animació.

- Arrossegarem l'arxiu Jump (32x32) a l'escena i fixeu-vos que no ens demanarà que guardem l'animació.
- Farem click a Window → Animation → Animation
- Farem click a Create sota el missatge “To begin animating Jump (32x32), create an Animator and Animation”.
- Guardarem l'animació amb el nom Jump



# Animació del personatge. JUMP

- Arrossegarem el sprite a l'animació



- Eliminem el sprite de l'escena
- Eliminem el controlador Jump (32x32) de la carpeta FrogAnimation

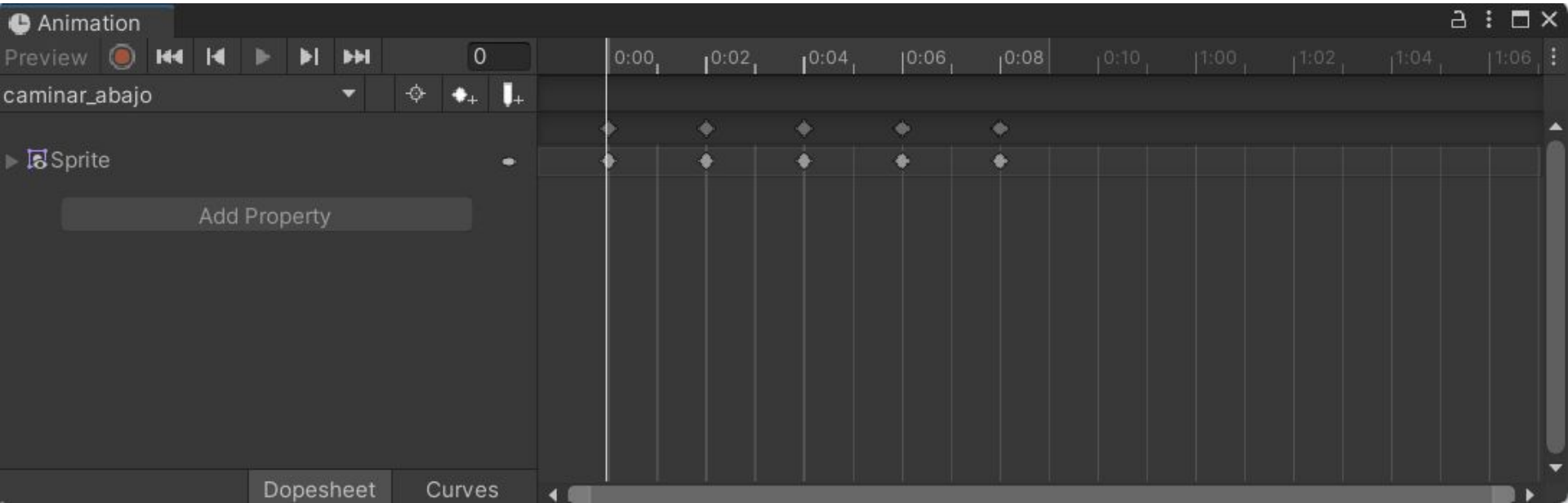
# Exercici lliure

Realitza l'animació del salt. Tingues en compte les següents consideracions:

- En el controlador d'animacions, podrem saltar si estem en repòs (IDLE) o si estem corrent (RUN), per tant, tindrem 4 transicions.
- Si estem corrent i saltem la variable run ha d'estar a false.
- Volem que faci l'animació de JUMP sempre i quan no estigui tocant el terra, no distingirem si salta o si cau.

# Modificació de temps d'una animació

Fent click a Window ➔ Animator i seleccionant l'animació podrem modificar el temps entre els diferents frames. Així com crear animacions des de zero.



# Animació del personatge (Blend tree)

# Blend tree

En el cas de tenir moltes animacions diferents d'un mateix personatge, l'esquema anterior se'ns complica molt i es fa difícil d'entendre i de programar. Una alternativa és utilitzar el Blend Tree.

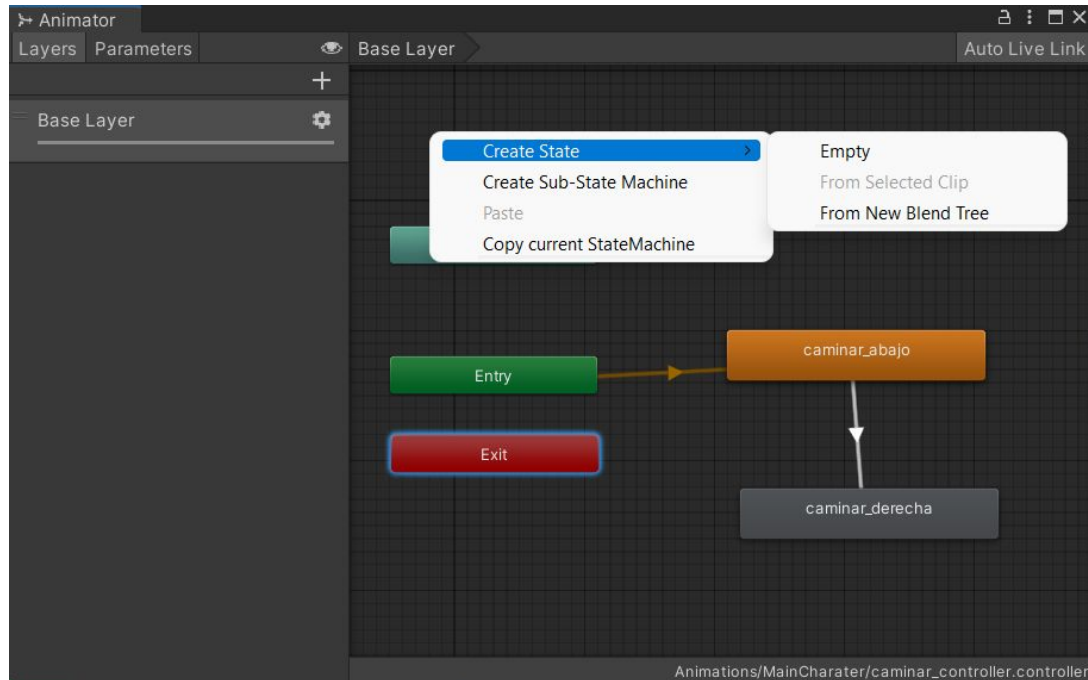
Normalment s'utilitza més en 3D que en 2D però és interessant fer-hi un cop d'ull.

[Documentació](#)



# Blend tree

Dins l'Animator farem click a Create State → From New Blend Tree

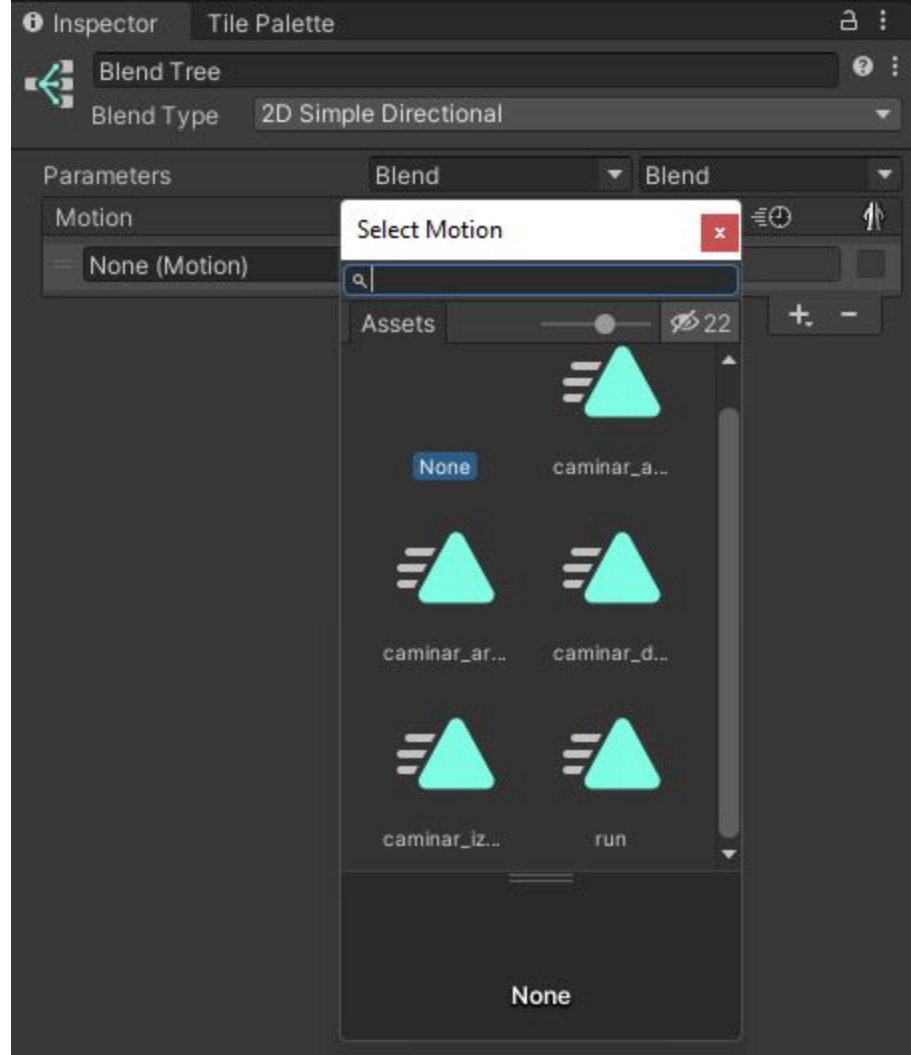


# Blend tree

Ara afegirem totes les animacions. Farem click al + (Add Motion Field) i seleccionarem l'animació.

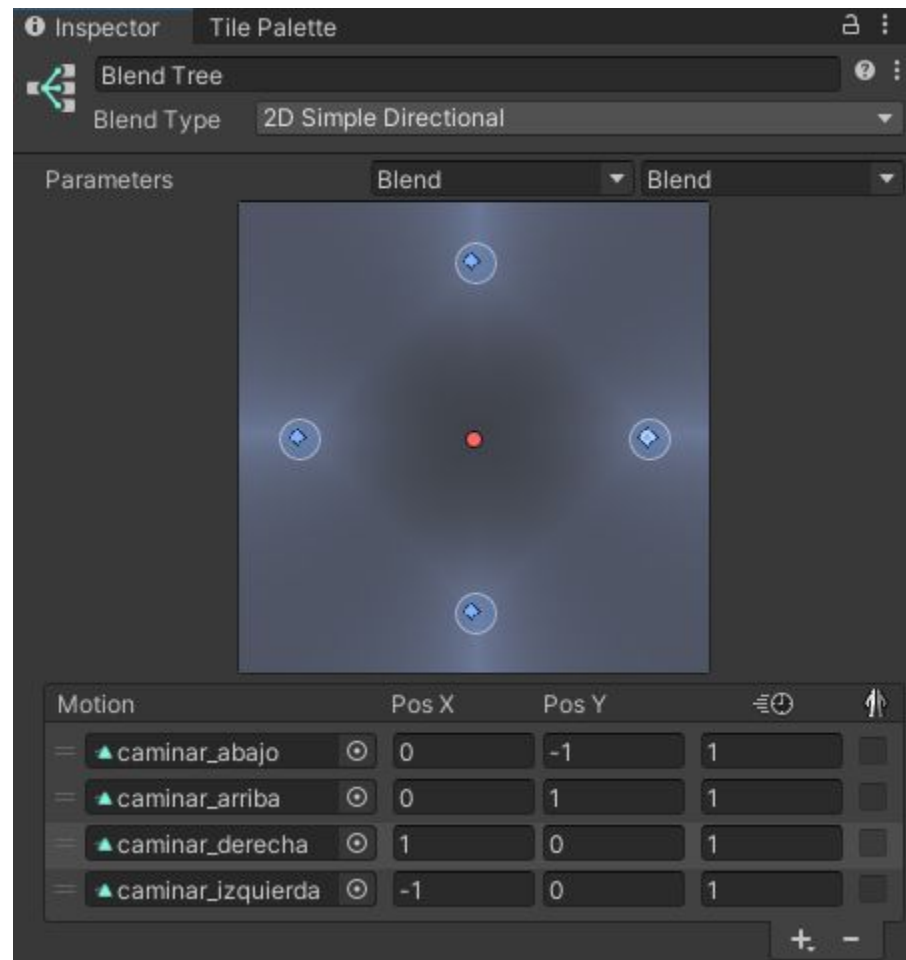
Alhora posarem els paràmetres tal i com nosaltres considerem.

Això ho farem per tantes animacions com tinguem del personatge



# Blend tree

Exemple d'una animació d'un personatge  
d'un videojoc Top Down



# Scripts amb Blend Tree

El scripting és semblant però d'aquesta manera el podem fer més desglossat.

Per una banda li afegim a l'usuari un script pel moviment i un altre per l'animació.

Analizem-los a continuació

# Scripts amb Blend Tree

```
public class MainCharacterMoveGame : MonoBehaviour{
    public Vector2 DireccionMovimiento => _direccionMovimiento;
    public bool EnMovimiento => _direccionMovimiento.magnitude > 0f;

    public float velocidad = 3;

    private Rigidbody2D _rigidBody2D;
    private Vector2 _direccionMovimiento;
    private Vector2 _input;

    void Awake(){
        _rigidBody2D = GetComponent<Rigidbody2D>();
    }

    private void Start(){
    }
```

# Scripts amb Blend Tree

```
private void Update() {
    _input = new Vector2(Input.GetAxisRaw("Horizontal"), Input.GetAxisRaw("Vertical"));
    if(_input.x > 0.1f){
        _direccionMovimiento.x = 1f;
    }else if(_input.x < 0f){
        _direccionMovimiento.x = -1f;
    }else{
        _direccionMovimiento.x = 0f;
    }

    if(_input.y > 0.1f){
        _direccionMovimiento.y = 1f;
    }else if(_input.y < 0f){
        _direccionMovimiento.y = -1f;
    }else{
        _direccionMovimiento.y = 0f;
    }
}

private void FixedUpdate() {
    _rigidBody2D.MovePosition(_rigidBody2D.position + _direccionMovimiento * velocidad *
Time.fixedDeltaTime);
}
}
```

# Scripts amb Blend Tree

```
public class MainCharacterAnimaciones : MonoBehaviour{
    private Animator animator;
    private MainCharacterMoveGame _personajeMovimiento;

    private void Awake() {
        animator = GetComponent<Animator>();
        _personajeMovimiento = GetComponent<MainCharacterMoveGame>();
    }
    void Start(){
    }

    void Update(){
        if(_personajeMovimiento.EnMovimiento){
            _animator.SetFloat("posX", _personajeMovimiento.DireccionMovimiento.x);
            _animator.SetFloat("posY", _personajeMovimiento.DireccionMovimiento.y);
        }
    }
}
```