SESSIÓ 2. (3)

Ampliar les capacitats del protagonista

5. Implementar més estats del protagonista: Saltar.

5.0 Recapitulació del procediment de gestió dels estats del joc

Hem vist que, per poder gestionar la infinitat d'esdeveniments que es produeixen en un joc, és molt útil implementar una **màquina d'estats** per seguir el comportament del Player (i, si cal, d'altres objectes del joc).

Es defineixen els **estats** en que pot trobar-se el Player, en funció dels comportaments que li hem assignat en dissenyar el joc. Una vegada establerts els estats possibles, definim també les **transicions** estre estats: per a cada estat (**estat actual**) establim quin **esdeveniment** és significatiu (té associat un tractament), quin és aquest **tractament** associat i quin és l'**estat destí** al qual es passa, després de realitzar el tractament associat.

Per exemple: a l'estat Idle (**estat actual**), polsar la fletxa a la dreta és un **esdeveniment** significatiu. Ho és perquè suposa realitzar un **tractament**: canviar d'animació. Al final, l'estat canvia de Idle a Left (**estat destí**).

Tot això ho esten implementant per mitjà d'scripts C#:

A l'script **PlayerStateController** detectem les entrades de l'usuari (de teclat o, en general, de qualsevol dispositiu) i decidim si determinen un canvi d'estat.

Utilitzem el mecanisme delegate+event de C# i , si hi ha canvi d'estat, es genera un event que indica el nou estat. Qualsevol dels altres objectes del joc poden assabentarse dels canvis d'estat del Player, simplement inscribint-se a la llista de Listeners de l'event.

De moment, amb l'script **PlayerStateListener** el Player pot detectar els seus propis canvis d'estat i fer les accions adequades i amb l'script **CameraController** la càmara principal (Main Camera) detecta els canvis d'estat per poder "enfocar" contínuament el Player.

5.1 Jump (1a part): Salt

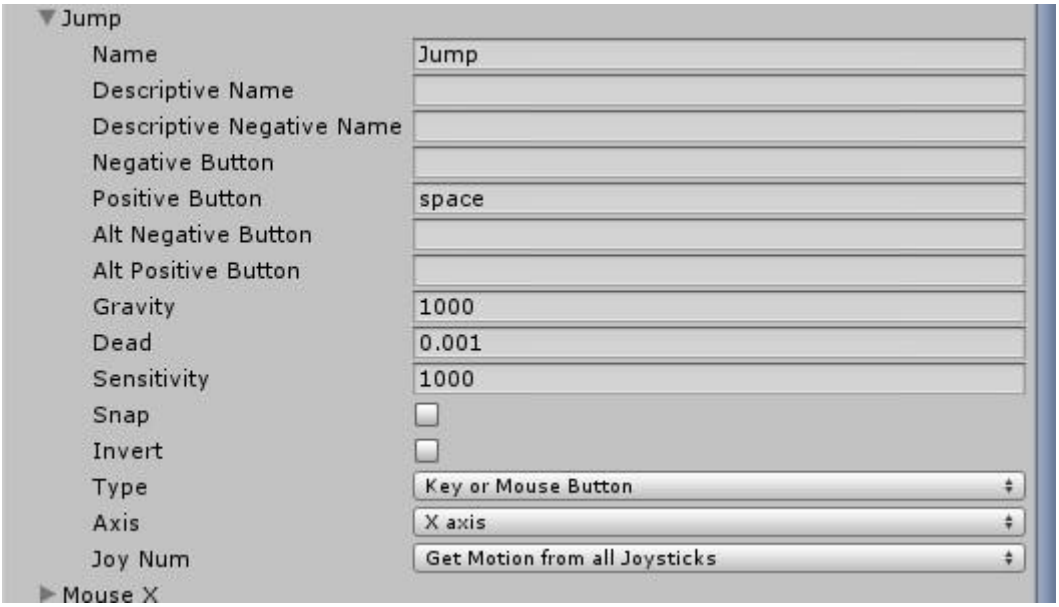
Anem a afegir una nova habilitat al Player: saltar quan l'usuari polsi la barra d'espai (és la tecla estàndard per fer-ho)

Per fer saltar al Player hem de realitzar dos tipus de tractaments:

1. Detectar l'entrada de l'usuari (per passar a estat **jump**)
2. Gestionar el canvi a estat **jump**.

5.1.1 Implementació del punt 1:

Si anem a Edit > Project Settings > Input podem veure els "axes" estàndard de Unity. Ja hem utilitzat l' "axis" **Horizontal** pels moviments rigth i left. Ara utilitzarem l' "axis" **Jump** per detectar la intenció de l'usuari de fer saltar al Player.



Veiem que la tecla associada per defecte és la barra d'espai.

Editem l'script **PlayerStateController** i, després del tractament de l'axis horitzontal, hi afegim la detecció d'entrada d'usuari en l'axis Jump:

```
float jump = Input.GetAxis("Jump");

if(jump > 0.0f)
{ if(onStateChange != null)
      onStateChange(PlayerStateController.playerStates
      .jump);
}
```

Comentaris al codi:

Simplement, quan hi ha entrada d'usuari en el axis Jump, generem un event de canvi d'estat cap a l'estat jump.

5.1.2 Implementació del punt 2:

Per començar el tractament del salt editem l'script **PlayerStateListener**.

Hi afegim el tractament de l'estat **jump** al mètode **onStateChange (C)** (que tracta els events de canvi d'estat) i també tractem l'estat **jump** en el mètode **checkForValidStatePair (D)**(que comprova que la transició sigui vàlida).

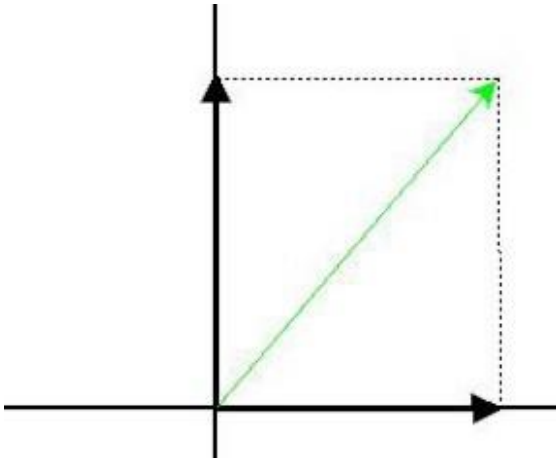
```
....
// -----------  AFEGIR amb la resta de variables públiques
(A) public float playerJumpForceVertical = 500f; public float
playerJumpForceHorizontal = 250f; ...
// -----------  AFEGIR amb la resta de variables privades (B) private
bool playerHasLanded = true;
...

(C) case
PlayerStateController.playerStates.jump:
      if(playerHasLanded)
      {
      // jumpDirection determina si el salt es a la dreta, esquerra o
            vertical float jumpDirection = 0.0f;
            if(currentState == PlayerStateController.playerStates.left)
            jumpDirection = -1.0f; else if(currentState ==
            PlayerStateController.playerStates.right) jumpDirection = 1.0f;
            else jumpDirection = 0.0f;

      // aplicar la força per fer el salt
            rigidbody2D.AddForce(new Vector2(jumpDirection *
playerJumpForceHorizontal,playerJumpForceVertical));
      //indicar que el Player esta saltant en l'aire
            playerHasLanded = false;
      }
break;
...

(D) case PlayerStateController.playerStates.jump:
            // Des de  Jump només es pot passar a  landing o a kill.
            if(newState == PlayerStateController.playerStates.landing
                  ||    newState ==
            PlayerStateController.playerStates.kill) returnVal =
            true; else returnVal = false;
      break;

...
```

Comentaris al codi:

En implementar els estats left i right hem vist una manera de desplaçar un objecte: modificar la seva posició per mitjà del **mètode Translate** del seu **component transform**. **Veure http://docs.unity3d.com/ScriptReference/Transform.Translate.html**

Ara veurem una altra manera de fer-ho: **aplicar una força al component rigidbody2D** de l'objecte que volem moure. El motor de físiques s'encarrega de simular el moviment que provocaria la força sobre l'objecte.

El vector força, en dos dimensions, té una component x (horitzontal) i una component y (vertical). La força resultant és la dibuixada en color.



Es defineixen dues variables públiques **(A)**

public float **playerJumpForceVertical** = 500f;

public float **playerJumpForceHorizontal** = 250f;

a partir de les quals es calculen els component de la força que s'aplicarà al Player.
Si el salt és massa gran o massa petit, es pot graduar adequadament el valor d'aquestes variables.

La **component vertical** serà sempre el valor de **playerJumpForceVertical**, que té un valor positiu (sempre és "cap amunt").

La **component horitzontal** serà **playerJumpForceHorizontal** multiplicada per **jumpDirection.** El valor de **jumpDirection** depèn del moviment actual del Player**:**

- Si camina cap a la dreta, val 1 i s'obté una força resultant com la de la imatge anterior.
- Si camina cap a l'esquerra, val -1 i la component horitzontal anirà cap a la part negativa de l'eix x. La resultant serà la simètrica a la de la imatge anterior respecte de l'eix y.
- Si està quiet, val 0. Per tant, no hi ha component horitzontal i la força és completament vertical cap a amunt. La força s'aplica per mitjà del mètode rigidbody2D.AddForce.

Veure http://docs.unity3d.com/ScriptReference/Rigidbody2D.AddForce.html

També s'ha de notar que tot aquest tractament només es fa si el Player està a sobre d'una plataforma (indicador **playerHasLanded(B)**). Així, quan apliquem la força, indiquem que no està sobre la plataforma (playerHasLanded = false) ja que és a l'aire a consequència de la força aplicada. **La gestió de l'indicador playerHasLanded s'ha de completar en el tractament del final del salt**.

Exercicis de la 1a part

Executar el joc tal com està en aquest moment, amb els últims canvis.

1. Provar els moviments a esquerra i dreta. Han de funcionar igual que abans. Si no és així, revisar el codi i veure què falla.

2. Provar el salt (barra d'espai). Veiem que el Player salta una vegada i el joc es bloqueja. **¿A que es degut ?.** Mirar el codi i **explicar raonadament perquè no va** el salt ni tampoc els desplaçaments horitzontals. (L'objectiu és acabar d'agafar familiaritat amb l'estructura del codi).