Me gustaría empezar a comentar el apartado sobre la interfaz con 2 aspectos que me han parecido realmente útiles en esta práctica.

Estoy hablando de los ‘spritesheets’ y las animaciones de los personajes. He comprobado que Internet cuenta con numerosos recursos para añadir distintos ‘sprites’ a nuestro juego. En este caso, UNITY cuenta con una herramienta que nos permite dividir un documento png con fondo transparente en cada una de las imágenes dibujadas. Así sólo nos tenemos que preocupar de descargar un archivo de sprites en png.

También he añadido varias animaciones para dar movimiento al personaje principal y a algún elemento del escenario.

Estos dos aspectos se añaden y modifican desde la propia interfaz. La primera vez cuesta entenderlo, pero después de dos horas equivocándote acaba saliendo solo.

También hemos aprendido a anidar distintos animaciones. Se escoge una animación por defecto, y a partir de ella se dirige haci las distintas animaciones.

Para dar continuidad a este comentario, voy a pasar a explicar el uso de ‘Tilemaps’. Usar un Tile map puede ahorrarte varios dolores de cabeza con algunos prefabs, además de permitir resultados asombrosos. En resumen se pueden sustituir 30 bloques de plataformas que conforman un nivel, por una capa ‘Tilemaps’ donde ‘pintamos’ los bloques directamente desde una ‘spritesheet’ previamente formalizada. Tiene multitud de opciones que se pueden cambiar para lograr un acabado a la medida de cada diseñador.

Hasta aquí he hablado del apartado más gráfico que prácticamente no necesita código fuente para funcionar.

Otra cosa son por ejemplo los movimientos del ‘Player’. Estoy hablando del movimiento sobre el eje X y el salto del personaje. El movimiento se logra con una línea que recoge el movimiento sobre el Input ‘horizontal’ mientras que el salto se consigue mediante un vector de tipo velocidad, que impulsa puntualmente al player en el eje Y. Este código funciona a veces sí y a veces no. Lo mas acertado, que se ha comentado en clase, consiste en hacer que el ‘player’ distinga entre un collider ‘in’ y un collider ‘out’ que dicta si el jugadaro entra en la colisión o por el contrario sale de la colisión. En el momento en que el jugador salga de la colisión, y siempre que no esté chocando con la cara inferior de una plataforma, entonces se producirá un unicao salto del personaje.

Si el ‘Player’ cae al fondo de la pantalla, chocará contra un objeto vacío de la misma longitud que el escenario de juego. Entonces, en el momento de activar el collide, el script indica al juego que el ‘Player’ tiene que perder una vida y ser recolocado en el principio del nivel.

Los enemigos se mueven entre waypoints, que pertenecen a un array de waypoints. Cuando el enemigo se encuentra a una distancia reducida, entonces cambia de trayectoria.

Para mantener algunas características del ‘Player’ en los cambios de escena, he utilizado un Singleton del GameManager. En el GameManager se guardan las variables que no se reinician al principio de un nuevo nivel y que se deben mostrar en el UI gracias al Scenecontrolles de cada escena. Debemos tener cuidado de comprobar que el GameManager no se destruya en los cambios de escena.

Para terminar sólo quiero hablar del componente Cinemachine, que resulta una herramienta increíble para crear una cámar semiinteligente. Gracias a todas las opciones disponibles en UNITY, disponemos de una cámara que sigue al personajey que se ciñe a los márgenes del escenario. Es posible que el Cinemachine sea la herramienta imprescindible para crear la ilusión de juego interactivo en el sencillo mundo del 2D.