Project_CHANDRA

Definición variables de control de los enemigos

Fecha revisión	Autor
6/03/2013	Maxi

- **active**: bool (Enemy.cs)
  - Indica si el enemigo ha sido activado.
  - Esta variable se pone a true cuando el contador de *Level* correspondiente a su tiempo de spawn llega a 0. (todos los enemigos se crean al principio de la carga de un nivel y se añaden a una lista de enemigos *List <Enemy> enemies*, y se activan uno por uno cuando les corresponda).
  - Se utiliza en los métodos *Update* y *Draw* de la clase ***Game***, de manera que solo se actualiza su lógica y sólo se dibujan los enemigos que tienen esta variable a **true**.

- **colisionable**: bool (Enemy.cs)
  - Indica si el enemigo es colisionable o no.
  - Esta variable se utiliza para comprobar las colisiones con las balas o con el jugador.
  - Es necesaria, aparte de ***active***, para, por ejemplo, cuando el enemigo ha sido alcanzado por una bala y explota, ***active*** tiene que seguir estando a *true* (para que se siga dibujando la explosión, y avanzando la animación), pero en cambio, no tiene sentido continuar calculando colisiones sobre el, ya que ha sido eliminado.

- **animActive**: bool (Animation.cs)
  - Indica si una animación está activa o no.
  - Es necesaria para desactivar el sprite de un enemigo cuando se ha ejecutado una animación dada y después no se tiene que pintar nada (*animActive* se pone a *false*).

- **erasable**: bool (Enemy.cs)
  - indica si un enemigo ha sido derrotado y su función en la partida ha terminado.
  - Esta variable es necesaria aparte de **active**, ya que los enemigos pueden permanecer con *active* a *false*, pero no deben de ser eliminados del juego (como cuando se crean en *Level*).

Ejemplo:

1. Un enemigo nuevo *EnemyWeak* se crea en la clase ***Level*** y se añade a la lista de enemigos (*List <Enemy> enemies*) que luego se utiliza en ***Game***. En principio, la

variable **active** del enemigo permanece a false, **colisionable** estará a *true* y **animActive** a *false* pero es indiferente, ya que no se llama al *Update* ni al *Draw* del enemigo (mientras *active* esté a *false*).

2. Tras el tiempo establecido, se "activa" al enemigo, poniendo su variable **active** a *true*. A partir de aquí, ***Game*** llamará al *Update* y al *Draw* del enemigo. Además la animación se irá actualizando y se comprobará el *Collider* del enemigo.
3. Suponemos que un enemigo ha sido alcanzado por una bala que ha eliminado toda su vida, en el método *damage* de *Enemy*, si *life <= 0*, se pone a false la variable **colisionable** (a partir de aquí ya no se actualizará el *Collider*).
4. Además, hay que indicar a la clase padre *Animation* la nueva animación que tiene que ejecutar (en principio una de muerte / explosión del enemigo), esta animación puede ser diferente en función del tipo de enemigo, por lo tanto hay que especificar qué animación hay que ejecutar en la clase hija de *Enemy* correspondiente al enemigo en concreto, por ejemplo *EnemyWeak*. Esto se hace en el método *damage* de *EnemyWeak* (que antes habrá llamado al de la clase padre, punto 3 del ejemplo), se chequea si *life <= 0* y se llama al *setAnim(int X,-1)* de la clase padre *Animation*.
5. En *setAnim(int X,-1)*, X sería el número de la animación de muerte, y el -1 sirve para indicar a *Animation* que después de ejecutar la fila de la animación X la variable **animActive** se tiene que poner a false.
6. Una vez se ha terminado de ejecutar la animación de muerte y **animActive** esté a false, podemos suponer que el enemigo está muerto y ya debería desaparecer del nivel, entonces en *Enemy* ponemos **erasable** a *true*, y en la siguiente iteración del *Update* de **Game**, al comprobar que el enemigo tiene esta variable a *false* se eliminará de la lista *enemies*.