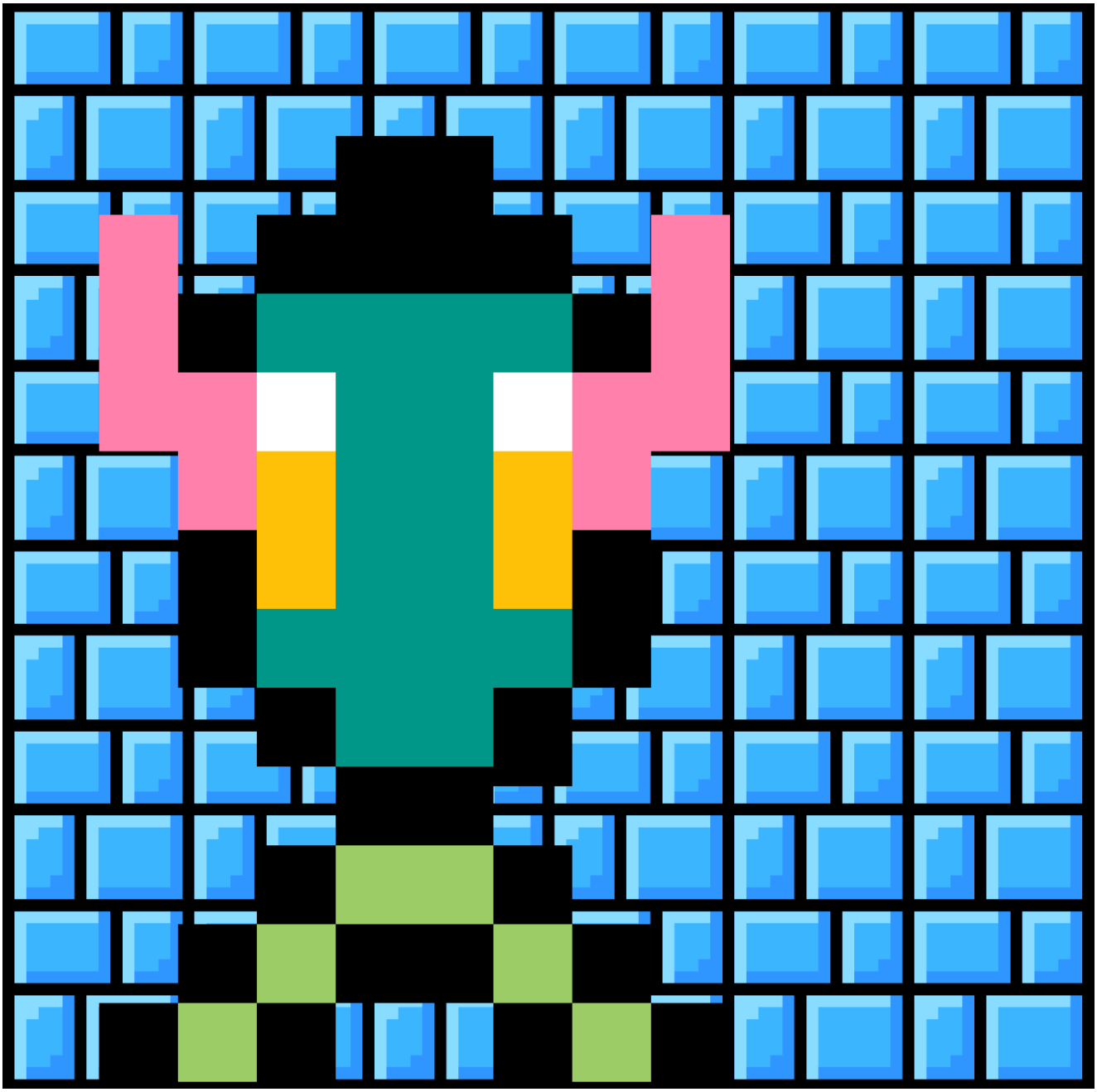
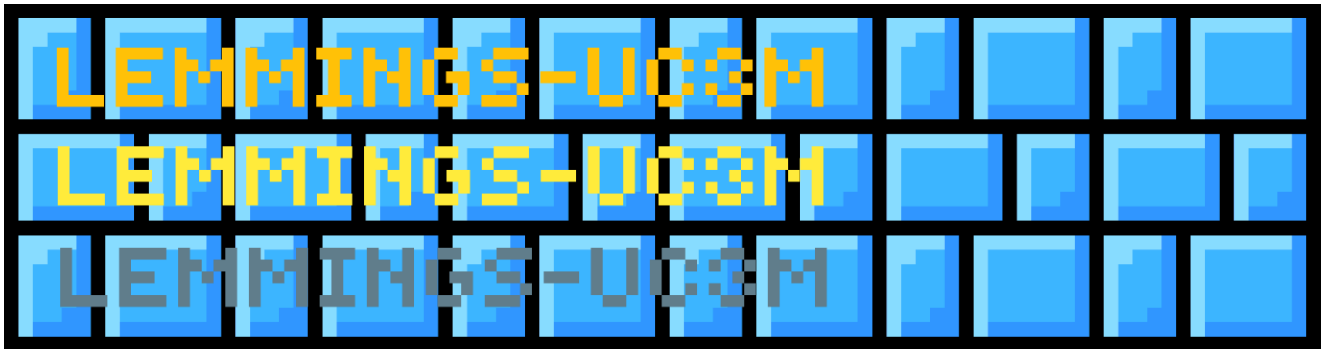




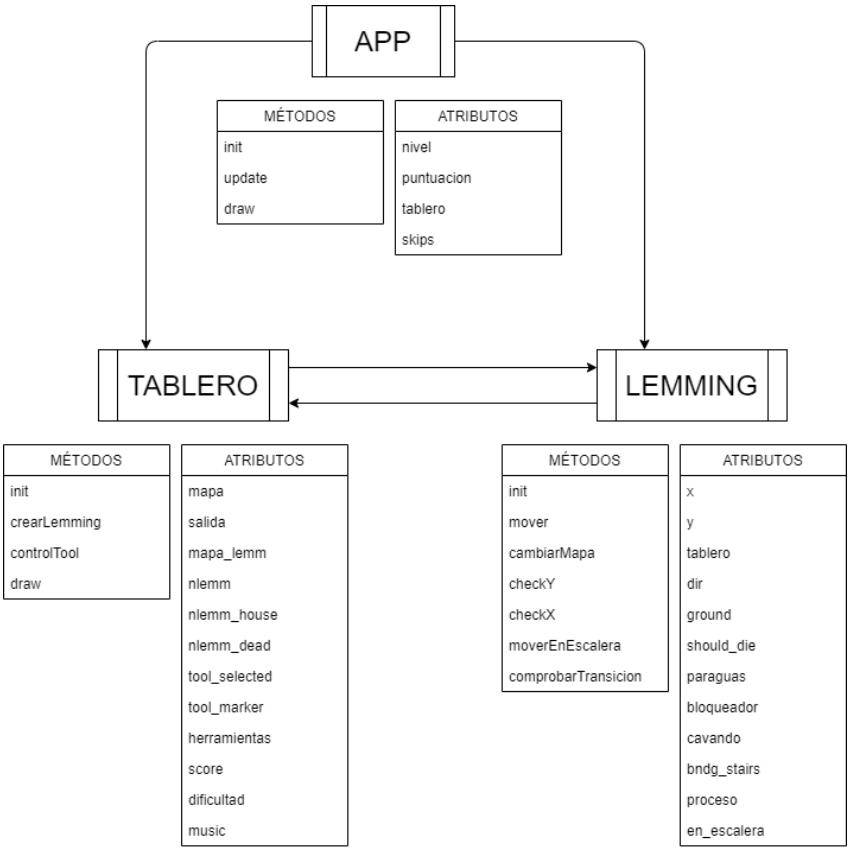
Prod. Alejandro Jiménez y Alberto Molina

ÍNDICE

RESUMEN

La siguiente memoria tiene como intención explicar nuestro programa para el trabajo final de programación en nuestro primer cuatrimestre del Doble Grado de Ingeniería Informática con ADE en la Universidad Carlos III de Madrid. Buscamos exponer el proceso de creación de nuestra versión del videojuego “Lemmings”, explicar el diseño de clases con sus métodos y atributos, los algoritmos y cómo funciona el programa en general. Ha sido un proyecto interesante nos ha hecho resolver problemas y utilizar un pensamiento más esquemático y propio de futuros ingenieros

DISEÑO DE CLASES



Hemos aprendido que este paso es uno de los más importantes en la construcción de un buen programa y vale la pena ponderarlo antes de lanzarse a programar. Inicialmente, optamos por más clases teniendo además de las de ahora plataformas y herramientas, pero con el desarrollo del programa decidimos recortarlo. De la misma manera, pensamos que era necesario el uso de los métodos *draw* y *update* en cada clase, finalmente decidimos obviar ambos métodos.

Como se ve en el diagrama la clase que controla todo es **APP**, esta es la responsable de crear los tableros, mover a los Lemmings y controlar las interacciones con el usuario. Guarda el tablero que está en juego como un atributo, así puede acceder a todos los métodos y atributos de este, que incluye a los Lemmings. El método *update* es el único en todo el programa ya que buscamos simplificarlo desde el principio y el *draw* solo se ocupa de la puntuación y el nivel. El método *init* de App únicamente crea la pantalla, crea el primer tablero y llama a pyxel.run que ejecuta *draw* y *update*. Los atributos son el *nivel* en el que estas, la *puntuación* de la partida, el *tablero* que estas jugando y los *skips* que quedan.

La clase **TABLERO** es más complicada y extendida que App, los métodos principales son *init, crearLemming*, *controlTool* y *draw*, tiene muchos más atributos ya que hay más cosas que controlar. Empezando por los métodos, *init* es responsable de crear el mapa del tablero, situando la casa y la meta, de elegir el número de Lemmings total que habrá y de inicializar el resto de los atributos. A continuación, *crearLemming* se ocupa, mientras los Lemmings creados sean menos que los totales, de crear un objeto Lemming en la celda de la casilla. El método *controlTool*, como indica el nombre, mueve y coloca las herramientas y los marcadores de estas y el comportamiento depende de la tecla que recibe. El atributo *tool_selected* dirá si se está colocando una herramienta y cuando se presione la tecla correcta el método cambiará esa celda a una herramienta. Por último, *draw* pinta las plataformas, la cabecera, los Lemmings, las herramientas y la animación final.

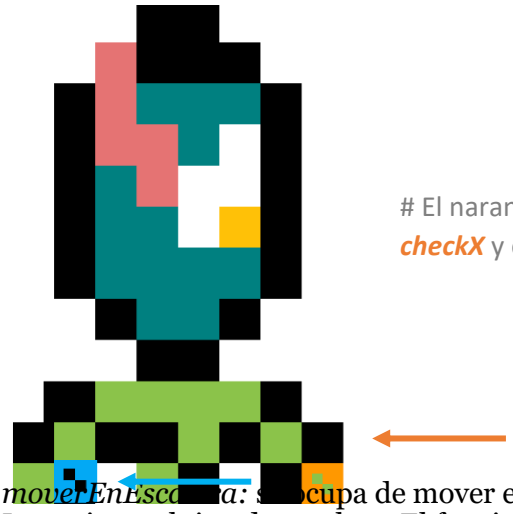
La necesidad de comprobar distintos datos o de tener que imprimirlos implica que cada objeto tablero tenga tantos atributos, en total son 10.

- *Mapa*: es una matriz 14x16 que guarda lo que hay en cada celda, desde suelo hasta palas. Los distintos valores en la matriz determinan que habrá en esas celda
    - 0: hueco
    - 1: suelo
    - 2: meta
    - 3: salida
    - -1: paraguas
    - -2: pala
    - -3: escalera sin construir hacia la derecha
    - -3.1: escalera construida hacia la derecha
    - -3.2: esquina que une las escaleras
    - -3.5: escalera sin construir hacia la izquierda
    - -3.6: escalera construida hacia la izquierda
    - -3.7: esquina, izquierda, que une las escaleras
    - -4: señal de bloqueador
    - -4.1: Lemming bloqueando
- *Salida*: guarda las coordenadas de la celda de salida y evita tener que recorrer la matriz entera para crear a los Lemmings
- *Mapa_lemm*: es una lista con todos los Lemmings y hace posible acceder a la información de estos desde App y de tener a todos ellos guardados en un solo sitio
- *Nlemm*: es el número de 10-20 aleatorio de Lemmings que se generan por Tablero
- *Nlemm_house*: el número de Lemmings que han llegado a la meta
- *Nlemm_dead*: los que han caído y muerto
- *Tool_selected*: un booleano, verdadero cuando hay un señalizador de herramienta en pantalla
- *Tool_marker*: una lista guarda las coordenadas del señalizador de la herramienta y el tipo

- *Herramientas*: es una lista con el número de herramientas disponibles en cada nivel
- *Score*: guarda la puntuación en ese nivel
- *Music*: se refiere a si queremos escuchar la música y los efectos de sonido en ese nivel

La última clase es **LEMMING,** tiene más atributos y más métodos que las otras dos y también se debe a la cantidad de información que tiene un Lemming y las necesidad de acceder desde otras clases. Empezando con los métodos:

- *Init:* el más simple de todos, solo inicializa los atributos correctamente
- *Mover:* primero llama a los dos métodos "check" y hace uso de la mayoría de los atributos de los Lemmings, para mover al Lemming por el tablero excepto por las escaleras
- *cambiarMapa:* con el atributo *proceso* cambia la matriz del tablero excavando o construyendo las escaleras. Se llama cada medio segundo para poder alargar la mano de obra de los Lemmings. El algoritmo de construir escaleras es más complicado y se explicará posteriormente.
- *checkY:* recorre la matriz entera hasta encontrar la celda donde está el **"pixel crítico A"**, se ocupa de decidir el atributo *ground* y *should_die* que dependen de si hay o no plataforma en ese pixel. Depende de la dirección porque el pixel cambia.
- *checkX:* de la misma manera recorre la matriz del tablero y mira que hay en la celda del **"pixel crítico B"**. Se ocupa de la dirección, de si está en casa y del resto de los estados que tiene el Lemming que se pueden acceder en el eje x. Diferenciar los pixeles de esta manera evita glichtes y facilita el uso de herramientas.



- *moverEnEscalera:* se ocupa de mover en las escaleras, depende de la dirección del Lemming y el tipo de escalera. El funcionamiento del algoritmo se explicará más tarde.
- *comprobarTransicion:* al mirar pixeles opuestos en *checkY* y *moverEnEscalera* la transición de escalera a plataforma necesita un método aparte porque hay un momento en el que según esos métodos el Lemming no está ni en escalera ni en suelo.

Los atributos de la clase Lemming son más sencillos:

- *x:* coordenada x
- *y:* coordenada y

- *tablero:* tablero en el cual está el Lemming
- *dir:* dirección del Lemming, 0 hacia la derecha, 1 hacia la izquierda
- *ground:* si hay suelo o no, 1 si, 0 no
- *should_die:* si cuando toque el suelo debería morir
- *paraguas:* si es en esa caída ha tocado un paraguas
- *bloqueador:* si está funcionando como bloqueador
- *cavando:* si ha tocado una pala y está cavando
- *bdng_stair:* si está construyendo escaleras
- *proceso:* en que parte del cambio del mapa (cavando/construyendo) está
- *en_escalera:* si está en una escalera

ALGORITMOS

APP está dividida en pequeños módulos de lógica que resetean a los Lemmings, crean un nuevo tablero, llama a los métodos principales de las otras clases y mueve las herramientas

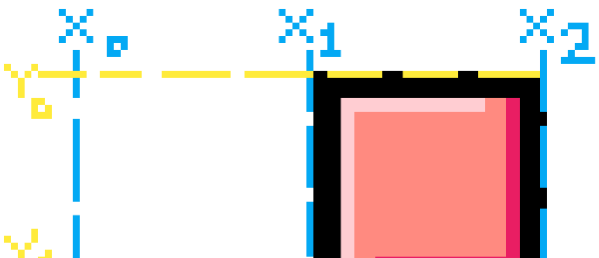
En TABLERO los algoritmos menos obvios están en el *init* y *controlTool*.

1. *Init*: en la creación del mapa inicialmente se crea una matriz de las dimensiones deseadas rellena de ceros excepto las paredes y el suelo. Posteriormente, se elige que niveles de altura tendrán una plataforma. Hemos decidido poner las plataformas pegadas a las paredes y las alturas pares estarán a la derecha mientras que las impares a la izquierda, evitando así problemas si se solapan. Por último, se escoge una de las plataformas y se sitúa encima de una la salida y en otra la meta
2. *controlTool*: el algoritmo se divide en tres partes, si se llama el método cuando *tool_selected* es Falso y si quedan suficientes herramientas de ese tipo crea el marcador. Si es verdadero comprueba que tecla hemos presionado, si es una de las herramientas que no hemos seleccionado devuelve el marcador al inicio, si es una flecha mueve el marcador y si es la tecla de la herramienta seleccionada coloca la herramienta

En LEMMING hace falta explicar los algoritmos de *cambiarMapa, moverEnEscalera y comprobarTransición*.

1. *cambiarMapa*: cada vez que se llama este método aumento el atributo *proceso,* el algoritmo tiene 2 partes principales,
    a. Cavando: si el proceso es 0 quita la herramienta, después cambia la celda inferior a suelo agrietado, hasta que 90 frames después rompe el suelo.
    b. Escaleras: si el proceso es 0 hacemos la celda de delante una escalera, después cada vez que aumenta el proceso miramos la celda de delante y arriba y mientras estas estén vacías se ponen escalera. Los *ifs* son algo intimidantes, pero el proceso hace que se avance en la dirección correcta y los enteros que se suman o restan sirven para ajustar el desfase de las celdas por la cabecera y la altura del Lemming.

2. *moverEnEscalera:* podemos dividir la posición de un lemming en 4 partes, dependiendo de la x e y respecto a la escalera, esto lo hacemos con el módulo 16 de la coordenada.



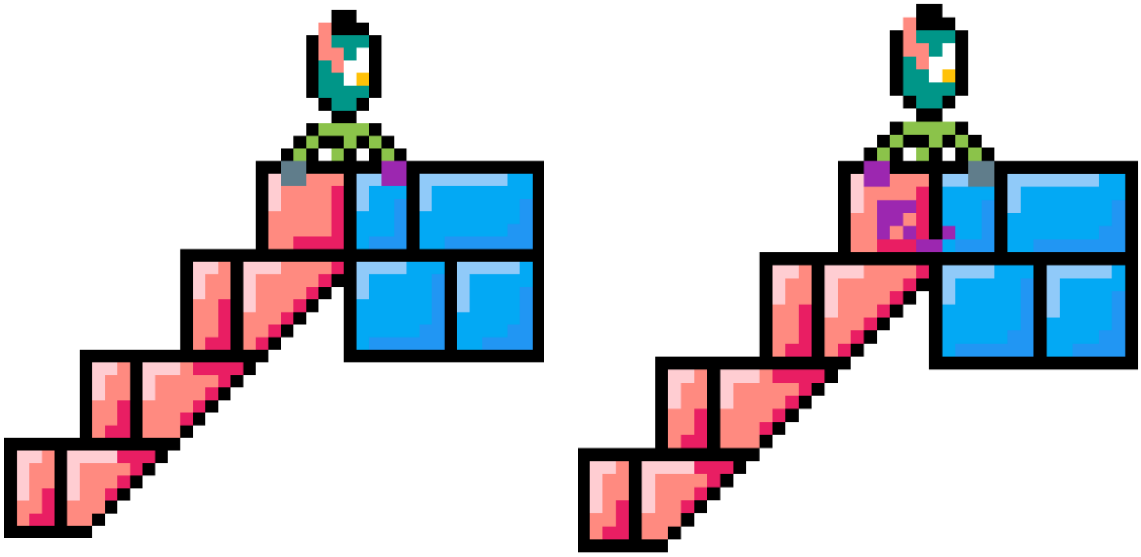
Como queremos que el Lemming se mueva ajustado a la escalera tendremos que sumar o restar a su coordenada

Primero decimos que un Lemming está en una escalera, *en_escalera = True*, si uno de sus pixeles críticos están en una celda escalera o transición de escalera. Por el ajuste si no está en escalera según el primer *if* miramos un pixel debajo ya que en el último tramo está encima de la escalera, pero no en la escalera.

El movimiento cambia con la escalera y con la dirección, pero en todos los casos se mira en que sector está la coordenada X y dependiendo de la Y se avanza en el eje X o Y. El único caso distinto es cuando bajan las escaleras, como queremos que puedan pasar por debajo para poder subir después comprobamos dos píxeles para decir si *en_escalera* es True. Miramos 3 píxeles por debajo para que solo este en escalera cuando este encima de la escalera.

3. *comprobarTransicion*: para resolver el problema que nombramos previamente cambiamos los pixeles que miramos, así hacemos que *ground* sea 1 y se comporte como si hubiese suelo y que ponga que está en una escalera.

En la primera figura el Lemmings se caería, pero el método hace que se comporte correctamente



FUNCIONAMIENTO

En el momento que se ejecuta el programa se crea un objeto App, este crea la pantalla, el primer Tablero y se llama a los métodos principales de Lemming y Tablero. El Tablero en juego pinta las plataformas y según se crean también pinta a los Lemmings.

Los Lemmings se mueven cada frame, comprobando el eje X e Y antes de hacerlo. Normalmente es necesario usar herramientas solo para que los Lemmings sobrevivan, pulsando la tecla A aparece el marker del paraguas y se mueve con las flechas. Para poner el paraguas se pulsa la tecla A de nuevo, si el Lemmings cae más de 2 celdas morirá. Las escaleras son otra herramienta muy útil, cuando movemos el marker podemos usar ENTER para rotar la escalera. Cuando un Lemming llegue empezará a construirla y los demás podrán usarlas como en el juego original.

Respecto a la construcción de las escaleras, hemos decidido poner que los Lemmings la puedan usar instantáneamente, porque así funciona en el juego original.

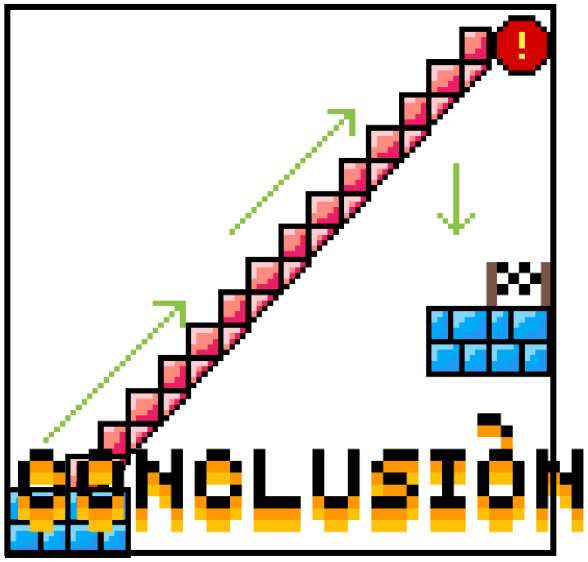
Por la forma en la que hay que crear las plataformas hay muchos niveles imposibles, por eso hemos puesto *skips*, 5 por partida, servirán para intentar otro nivel sea este imposible o no.

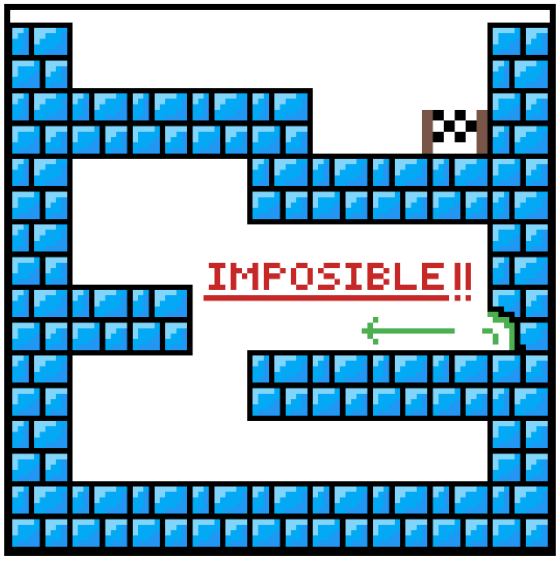
BOTONES

- A, S, D ,F: seleccionar y poner herramientas
- R: recargar los Lemmings de un nivel, reducirá la puntuación
- M: mutear el sonido por el resto del nivel
- T: pasar de nivel cuando han llegado todos a casa
- P: pasar a otro nivel, solo se puede 5 veces
- Ratón: quitar herramientas y des-*bloquear* un Lemming

Ejemplo de nivel imposible, pero colocando una herramienta en el camino de la escalera podemos hacer que lleguen a la meta

Ejemplo de nivel imposible por las plataformas





En conclusión, nuestra versión de "Lemmings" es un intento de aplicar los conocimientos adquiridos estos 4 meses en unas 800 líneas de código. Primero pensamos el diseño de clases, la implementación de los métodos y la relación entre las clases y los punteros que cada uno tiene con los demás. Inicialmente desarrollamos la plataformas y el movimiento de los Lemmings, después una a una implementamos las herramientas, ajustando los problemas y desarrollando los métodos que ya teníamos.

Los problemas que encontramos a lo largo de estas 3 semanas han sido suficientes, pero los principales fueron:

1. El movimiento de los Lemmings: desde el principio al mover los Lemmings en la caída pusimos que se moviesen de 3 en 3, de esta manera siempre se metían en las plataformas. Lo intentamos arreglar moviendo los pixeles que estudiábamos, pero se alargó y hizo mucho más difícil la implementación de las escaleras, hasta que hace poco nos dimos cuenta y ahora caen de 2 en 2, que se divide mejor entre 16.
2. Píxeles críticos, al principio pintábamos los Lemmings con círculos, pero después de pintar los Lemmings con pyxeleditor ajustar la anchura de los sprites resultó ser más difícil de lo que esperábamos.
3. Diseño de clases: desde el principio la clase HERRAMIENTA no tenía mucho sentido para nosotros, habíamos escuchado el término "herencia de clases", pero no lo habíamos visto todavía. Decidimos no crear la clase y solo cambiar la propiedad *mapa* del Tablero.

FEEDBACK: si volviésemos a hacerlo desde 0, tendríamos otro diseño de clases usando tal vez herencia para las herramientas. Incluimos un property porque es obligatorio, pero con nuestro diseño del programa no tiene ningún sentido. También cambiaríamos el diseño del programa en general para poder beneficiarnos de los properties y los setters. Por último, otra crítica que vemos en nuestro programa es el orden, podríamos haber ordenado y implementado de otra manera los métodos.