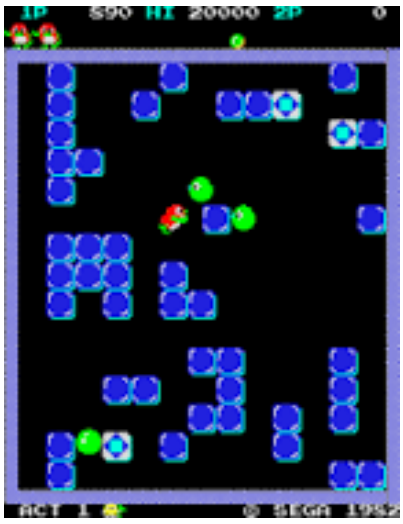
PRÁCTICA INDIVIDUAL

Desarrolla un videojuego de MSX. Para ello se utilizará únicamente C++ junto con SFML. Como entorno de desarrollo se debe utilizar VSCODE con CMAKE. La práctica se probará sobre **Ubuntu 16.04**

PENGO (SEGA, 1982)

SOBRE EL JUEGO

*Help **Pengo** to defeat all the Sno-Bees that go behind him while he crosses all the mazes. Squash them with the ice blocks or stop them temporally using the screen borders. Don't let them to use their sting on you!*



CARACTERÍSTICAS

La composición de los niveles del juego es parecida a los del Pac-Man: una pantalla única en forma de laberinto. Los enemigos de Pengo, los Sno-Bees, son parecidos a los fantasmas del juego de Namco. Los Sno-Bees vienen también en cuatro variaciones de color.

Un aspecto que diferencia Pengo del Pac-Man es que el laberinto es totalmente interactivo. Cada sección está representada por un bloque de hielo. El jugador usa el cursor del teclado y un botón para controlar a Pengo. Pulsando el botón hará que el bloque de hielo se deslice en la dirección en la que está mirando, hasta que choque con otro bloque o el muro, si el espacio que está directamente a continuación no está ocupado por otro bloque o muro. En cambio, si el espacio está ocupado, al pulsar el botón el bloque se romperá.

La meta del juego es acabar con todos los Sno-Bee de la pantalla, usando alguno de estos tres métodos:

- Deslizar bloques de hielos para aplastarlos
- Romper bloques que contengan huevos de Sno-Bee sin eclosionar
- Correr sobre ellos mientras están aturdidos por la sacudida del muro

Al principio de cada ronda, eclosionan un cierto número de huevos y se convierten en Sno-Bees, mientras que otros bloques parpadean indicando que contienen más huevos. A medida que el jugador destruye Sno-Bees, eclosionan más huevos para reemplazarlos. Aplastar varios Sno-Bees con un bloque da puntos extra. Los Sno-Bees pueden romper bloques para intentar llegar a Pengo. Pengo puede golpear el muro límite y hacer que vibre y, temporalmente, aturdir a cualquier Sno-Bees que esté en contacto con él; el jugador puede aplastarlo con un bloque o, simplemente, correr por encima de él para aplastarlo. El contacto con un Sno-Bee en otro caso cuesta una vida del jugador.

En cada ronda se marcan tres bloques como diamantes, y no se pueden romper. Al colocar estos bloques juntos, en horizontal o vertical, premia al jugador con puntos extra, y aturde temporalmente a todos los Sno-Bee activos. Además, provocará el parpadeo de los bloques que aún contengan huevos sin eclosionar.

Si el jugador elimina a todos los Sno-Bee en menos de 60 segundos, obtendrá más puntos calculados según el tiempo que haya tardado.

En cada ronda, hay un límite de tiempo de 2 minutos de inactividad por parte del jugador. Si se llega a ese límite (por ejemplo, el jugador aturde a los Sno-Bees evitando que le maten), todos los Sno-Bees, incluidos los que aún no han eclosionado, entran en modo de huida.

El proceso de huida del último Sno-Bee es el siguiente:

- Cuando quedan dos Sno-Bees, se pone un contador interno a 0
- Cuando queda un solo Sno-Bee, el contador se va incrementando; cuando llegue a 12 segundos, el Sno-Bee huirá
- En cambio, si al quedar tres o cuatro Sno-Bees, aplastamos con un bloque a todos ellos menos a uno (no pasamos por el caso de que existan dos Sno-Bees activos), el último Sno-Bee huirá inmediatamente (siempre y cuando llevemos en la ronda más de 12 segundos).

El juego incluye un total de 16 rondas, con distinto número de Sno-Bees y de dificultad. Al finalizar la decimosexta, se vuelva a la primera. Al final de cada ronda par, se muestra una de seis animaciones intermedias.

La generación de los laberintos es totalmente aleatoria, pero como el generador de números es muy simple, se producen repeticiones en los patrones de generación, pero tiene la ventaja de que la posición de los huevos y de los diamantes siempre es la misma.

¡¡¡Atención!!!, ante cualquier duda de funcionalidad se debe implementar la versión de pengo de MSX, disponible aquí: https://www.file-hunter.com/MSX/index.php?id=pengo

Texto extraído de https://es.wikipedia.org/wiki/Pengo_(videojuego)

Requisitos Mínimos (para aprobar)

- Se deben implementar como mínimo dos niveles. No es necesario que sean generados aleatoriamente.
- Añadir todas las animaciones donde se encuentren en el juego. No se requiere usar los mismos sprites del juego aunque es recomendable. Se puede jugar aquí https://www.file-hunter.com/MSX/index.php?id=pengo
- Implementar la siguiente funcionalidad del juego original:
    - Capacidad de empujar/destruir bloques
    - Pengo se desplaza por el entorno con las condiciones del juego original
    - Pengo puede eliminar Sno-Bees adultos mediante bloques
    - Si pengo elimina a todos los Sno-Bees se pasa al siguiente nivel
    - Los Sno-Bees aparecen en el entorno para suplir a los que se eliminen como en el juego original. Sin embargo, no es necesaria la fase de huevo.
    - Pengo dispone de tres vidas. Muere si colisionó tres veces con algún Sno-Bee
- Obligatoriamente se deben utilizar las siguientes teclas. La no implementación de la funcionalidad descrita a continuación supondrá la calificación de suspenso de la práctica:
    - ESC: salir del juego
    - Teclas del cursor para desplazar al jugador
    - Espacio para empujar
    - G modo dios (inmortal)
    - X muerte del personaje y reinicio del nivel
    - N para pasar al siguiente nivel sin necesidad de eliminar a los enemigos.
- No se requiere menú, ni presentación, ni pantalla de final de partida. No se requiere sonido. En cuanto nos maten se debe volver a reiniciar el juego.
- Se puede gastar *tiled* para diseñar los mapas, pueden estar en el código se pueden generar aleatoriamente.
- El juego se ejecutará en ventana y no pantalla completa. Se podrá utilizar la resolución que se considere oportuna para el juego pero nunca mayor de 1080x720.

Notable

- Añadir los bloques estrella y su funcionalidad asociada
- Añadir el margen límite y su funcionalidad de aturdir
- Se pueden eliminar huevos de Sno-Bees
- Los Sno-Bees pueden romper bloques para llegar hasta Pengo
- Implementar el HUD del juego

SOBRESALIENTE

- Generar laberintos al azar
- Implementar el proceso de huida de los Sno-Bees
- Transición entre pantallas
- Implementar el bucle tipo 3 (mov. por velocidad y frecuencia update < render) o tipo 4 (interpolado). El juego debe ir igual de rápido independientemente de la velocidad del equipo.

CUESTIONES SOBRE LA PROGRAMACIÓN DEL JUEGO

- Se debe desarrollar en C++/CMAKE/VSCode/SFML sin ninguna librería adicional a excepción de parsers de XML/Json para la lectura de este tipo de ficheros (por ejemplo tinyXML...).
- El juego debe funcionar en ordenadores del estilo de los de la EPS sobre **Ubuntu 16.04**.
- No debe requerir recursos exagerados (CPU, GPU, RAM) para el tipo de juego que es.

PRÁCTICA INDIVIDUAL Y SISTEMA ANTICOPIA

- Esta práctica es individual y por tanto no se permite compartir código ni hacerla en grupo. Se verificará mediante sistema anticopia el código fuente entregado.
- No se permite el uso de ninguna librería externa a excepción de parsers de XML o Json. No se permite compartir NADA de código.

Cualquier práctica donde se detecte copia será automáticamente suspendida y se pasará la información a la EPS para las sanciones que correspondan.

DETALLES Y ENTREGA

Esta práctica es individual y por tanto no se permite compartir código ni hacerla en grupo. Se verificará mediante sistema anticopia el código fuente entregado.

La fecha de entrega es el miércoles 22 de Abril de 2020 hasta las 23:55 mediante Moodle, no siendo prorrogable dicha fecha.

La entrega deberá cumplir los requisitos enumerados a continuación. **En caso contrario se evaluará como suspensa**. Por favor, verificar que los cumplís antes de entregar:

- Se debe entregar un ZIP. No se aceptarán otro tipo de compresores.
- El zip contendrá el proyecto de vscode completo. Se pueden eliminar los ejecutables y los ficheros ".o" ya que se realizará una compilación para cada una de las prácticas entregadas.
- Recomiendo verificar que el proyecto compila en una máquina distinta a la vuestra (puede ser en una máquina virtual) para corregir posibles fallos en la entrega (ficheros no incluidos...).

En el mismo nivel (donde se encuentra el fichero '.code-workspace') debe existir un fichero "leeme.txt" que contendrá cualquier aclaración al respecto de la práctica. No se leerá otro fichero distinto ni otro formato distinto al txt.

APÉNDICE I: CMAKE Y VSCODE

CMAKE

Para gestionar el proyecto e incluir SFML utilizaremos CMAKE. CMAKE requiere la existencia de un fichero "CMakeLists.txt" donde se especifican directivas para indicarle que librerías vamos a utilizar, que ficheros se van a compilar, que directorios debe ver…

En la plantilla de la práctica se adjunta un fichero de ejemplo "CMakeLists.txt". Este depende de si SFML es >= 2.5 o no. Se adjuntan las dos versiones (la versión de 2.4 también va en 2.5 por ahora). **Este código se encuentra ya integrado en la plantilla proporcionada para la práctica**

cmake sfml 2.5

```
cmake_minimum_required(VERSION 2.8)
project(prueba)
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++1y -Wall -Wextra")

find_package(SFML 2.5 COMPONENTS graphics audio REQUIRED)
add_executable(prueba main.cpp)
target_link_libraries(prueba sfml-graphics sfml-audio)
```

cmake sfml 2.4

```
cmake_minimum_required(VERSION 2.8)
project(prueba)
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++1y -Wall -Wextra")
add_executable(prueba main.cpp)
set(CMAKE_MODULE_PATH "${CMAKE_CURRENT_LIST_DIR}/cmake_modules")
find_package(SFML REQUIRED system window graphics network audio)
if (SFML_FOUND)
    include_directories(${SFML_INCLUDE_DIR})
    target_link_libraries(prueba ${SFML_LIBRARIES})
endif()
```

Hay que notar que para agregar otras rutas o fichero a compilar se deberá modificar levemente este fichero. Recomiendo algún tutorial introductorio sobre CMAKE para tal efecto.

CONFIGURACIÓN VSCODE

Este proyecto requiere la instalación y la configuración de VSCODE y CMAKE. A continuación, se explican algunas pistas para dicha configuración**. Notar que tanto el proyecto ABP como la práctica se compilará y probará en Ubuntu 16.04**. No obstante, se especifica también la configuración en Mac, ya que tenemos asignado un laboratorio con este operativo:

Mac

- VSCode. Dentro de el mismo se requieren los siguientes plugins. Estos se buscan en el apartado de plugins de la barra de la izquierda (la L formada de 3 cajas)
  - @id:ms-vscode.cpptools+
  - @id:vector-of-bool.cmake-tools

- Instalar las utilidades de línea de comandos (compilador) mac: `xcode-select --install`. Comprobar que tenemos instalado el compilador clang (`clang++ -v`)

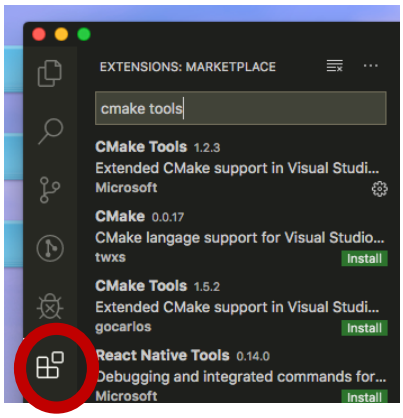
- Instalar brew: https://brew.sh/index_es
- Instalar los siguientes paquetes dentro de brew:
  - brew install cmake
  - brew install sfml
  - brew install gcc (solo si no se puede instalar clang o se desea gcc como compilador)

Linux

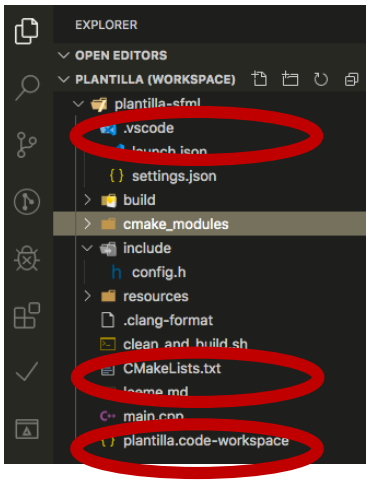
- sudo apt-get install libsfml-dev
- sudo apt-get install cmake
- sudo apt-get install clang
- VSCode. Dentro de el mismo se requieren los siguientes plugins. Estos se buscan en el apartado de plugins de la barra de la izquierda (la L formada de 3 cajas)
  - @id:ms-vscode.cpptools+
  - @id:vector-of-bool.cmake-tools

COMPILACIÓN EN VSCODE

Primero hay que instalar CPPTOOLS y CMAKE TOOLS en VSCODE:



Una vez instalado se debe abrir la plantilla proporcionada. Si desconocéis totalmente VSCODE se recomienda seguir algún tutorial de iniciación. Estas son las carpetas más destacables de la plantilla proporcionada.

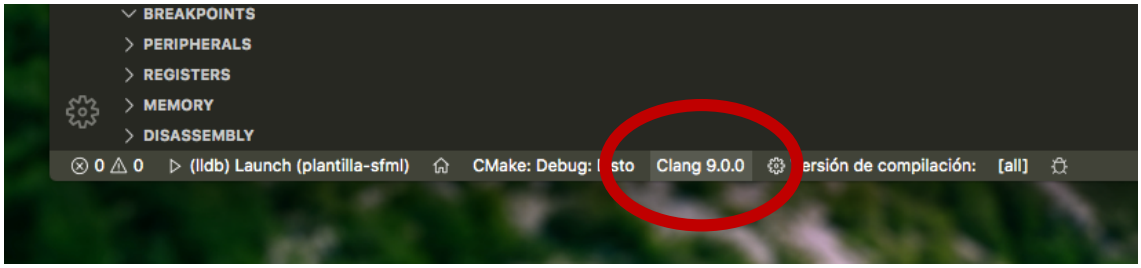


Carpeta con configuración de vscode

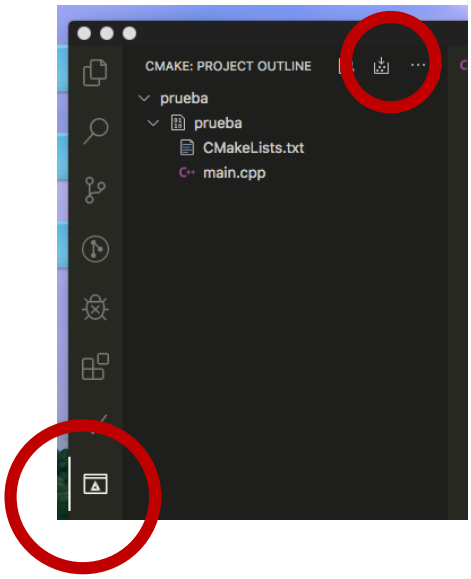
Fichero config CMAKE

Workspace a abrir

A continuación, hay que elegir el compilador a utilizar. Puede ser GCC o CLANG. Utilizar el mismo que gastéis para el proyecto ABP (todos los miembros del grupo deben utilizar el mismo)



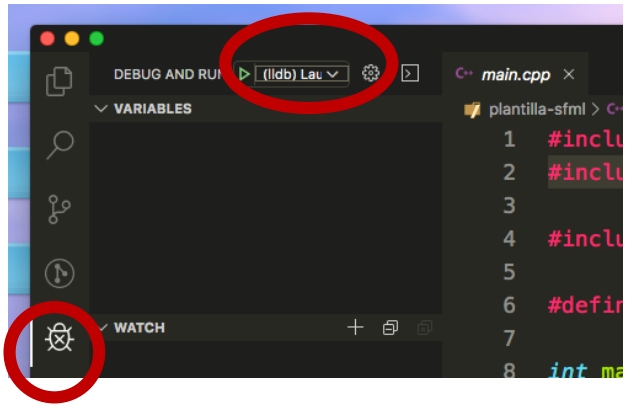
Ahora ya podemos compilarlo:



2) Pulsar el botón de compilar

1) Elegir la pestaña de CMAKE

Por último, podemos ejecutarlo en modo debug para depurarlo. Se pueden añadir breakpoints, visualizar el estado de las variables…



2) Seleccionar configuración debug y pulsar Play. Si no existe esta configuración se puede pulsar la rueda de configuración para crearla.

1) Elegir pestaña debug