M05 UF1 - P2 - Pacman Parte 2

Hoja de requisitos

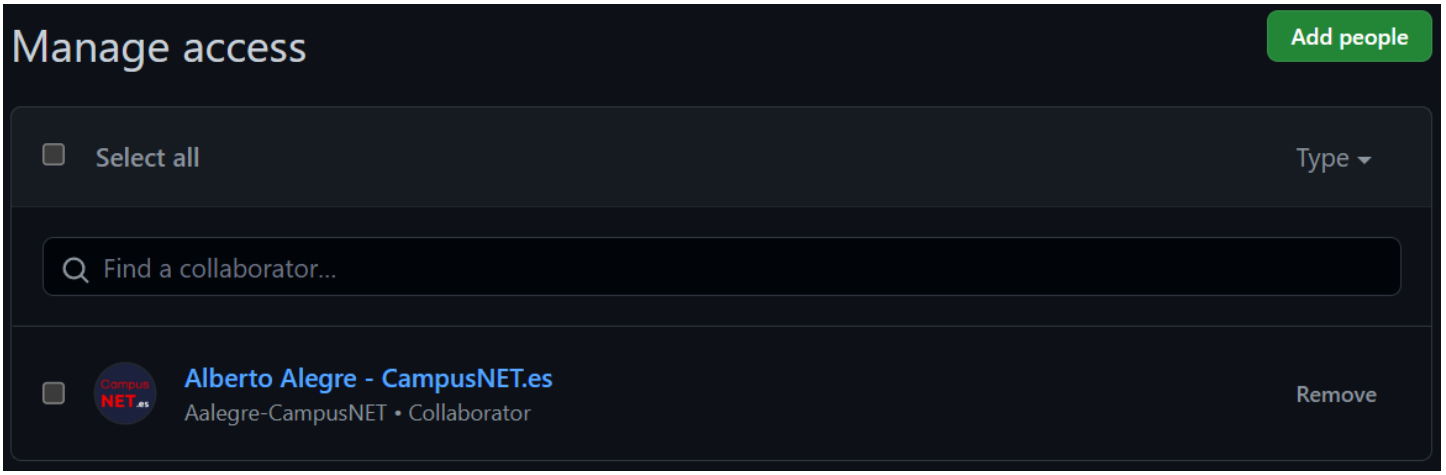
En este entregable, los alumnos por grupos realizarán un pacman semi-completo usando c++

Requisitos y entregable

Los alumnos deberán registrarse en un **excel de grupos** previamente a hacer la entrega. Incluidos aquellos alumnos que quieran realizar el trabajo en solitario.
El entregable requerirá de 2 partes. Si alguna de estas partes no se entrega, o no es accesible por el profesor, afectará negativamente a la nota.

Repositorio de GitHub

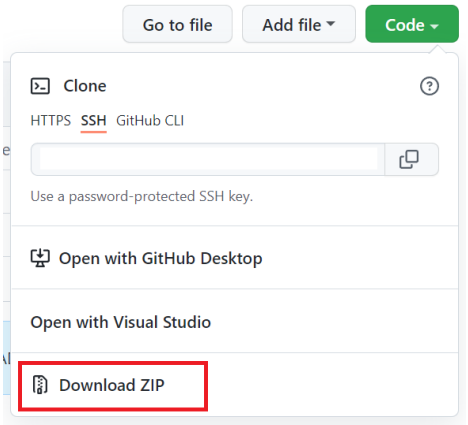
En el comentario de la entrega se tiene que añadir un enlace al repositorio donde se ha realizado el proyecto en Unity. Si el repositorio es privado, se tendrá que mandar una invitación al usuario https://github.com/Aalegre-CampusNET
La pestaña de colaboradores debería aparecer así, si se ha invitado a la cuenta correcta:



Si el profesor ve algún cambio en el repositorio, fuera de la fecha de entrega, se suspenderá la entrega con un 0.

Zip

Se comprobará que todos los miembros del grupo hayan entregado un .zip descargado del repositorio con el nombre correcto, el código se corregirá sobre dicho archivo.
Para descargar un .zip del repositorio, ir a la página del repositorio, seleccionar la rama adecuada, hacer click en el botón verde “code”, y seleccionar la opción de descargar en .zip.



Puntuación

Se pueden conseguir 13.5 puntos de 10, por tanto 3.5 puntos son extra. Si el alumno aprueba la sección básica, podrá acceder a los 3.5 puntos extra que le ayudarán a asegurar una buena nota.

Sección	Actividad	Punt	Descripción
GitHub - Básico	GitFlow	2.5	Se ha intentado seguir el formato de trabajo GitFlow
	Pull Request	1	Los cambios incorporados de Develop a Main/Master se han realizado con pull requests en vez de con merges simples.
	Etiqueta y release	0.5	Al finalizar el ciclo de desarrollo se ha etiquetado la rama Main/Master con un tag y marcada como release.
Desarrollo - Básico	Funcionamiento fantasmas	2	Los fantasmas se mueven, colisionan con las paredes y se imprimen en pantalla. Funcionan encapsulados en su propia clase, con sus propias funciones.
	Encapsulación player	1.5	El funcionamiento del player está completamente encapsulado en su propia clase
	Fantasmas en memoria dinamica	1	El número máximo de fantasmas depende de una decisión del jugador al principio de la partida, y estos van apareciendo cada 10 segundos hasta que hayan aparecido el número máximo de fantasmas
	Power ups	1	Se reparten unos pocos power ups por el mapa, al recogerlos se suman 10 puntos y el jugador durante 15 segundos podrá “matar” fantasmas, al matar a un fantasma se suman 50 puntos.
	Game over	0.5	Los fantasmas pueden “matar” al player, si el player muere 3 veces se pierde la partida
Desarrollo - Extra	Cargar mapa de archivo	2	El programa carga el diseño del mapa a través de un archivo que el usuario puede definir. Archivo de ejemplo: https://drive.google.com/file/d/15WTHZJUKrIio18o6QNvOMY5eyScTmZ4D/view?usp=sharing Formato del archivo: • La primera línea indica la anchura y la altura del mapa, separado por una coma • Las siguientes líneas indican el contenido del mapa ○ Símbolo **‘P’**: indica el spawn del jugador

			○ Símbolo **'E'**: indica el spawn de los enemigos ○ Símbolo **' '**: indica una casilla vacía ○ Símbolo **'#'**: indica una casilla de pared ○ Símbolo **'.'**: indica una casilla con un punto ○ Símbolo **'*'**: indica una casilla con un PowerUp
	Variables globales encapsuladas	1	No hay variables globales. Todos los tipos y las variables globales están almacenadas en un tipo definido por el usuario (struct o class)
	No hay valores "Hardcodeados"	0.5	Todos los valores, números, constantes, etc están almacenados en variables, constantes o macros