**Inteligencia Artificial para Videojuegos**

***Práctica 4: Frente de guerra zombie***



*Grupo 07*

*Alejandro Ortega Álvarez*

*Borja Cano Álvarez*

Problema a resolver

* La propuesta es representar en Unity un escenario o mapa (mediante casillas) donde se sitúa un agente y (opcionalmente) aliados que ayudan al agente para combatir una oleada de zombies.



* Para ello se piden los siguientes requisitos:

1. Un tablero de 6x12 (posiblemente modificable).
2. Al empezar, se sitúan el agente y el refugio (en este caso es una nave).Éstos se sitúan en las casillas 0,0 y 0,1.
3. Después el usuario puede poner cualquier cantidad de aliados y zombis distribuidos por el resto del mapa (hasta el límite que haya puesto, en este caso se pueden máx. 5 aliados y 20 zombies)
4. Una vez habiendo colocado las entidades como se desee, dándole al botón comenzar, empezará el simulador de batalla por turnos (los aliados no participan pero ayudan a que el agente se haga con la victoria).
5. El modo noche se puede activar y desactivar en cualquier momento de la partida e influye en el desarrollo de la misma.
6. El juego acaba si el agente muere, si vuelve a la nave o si consigue acabar con todos los zombies.

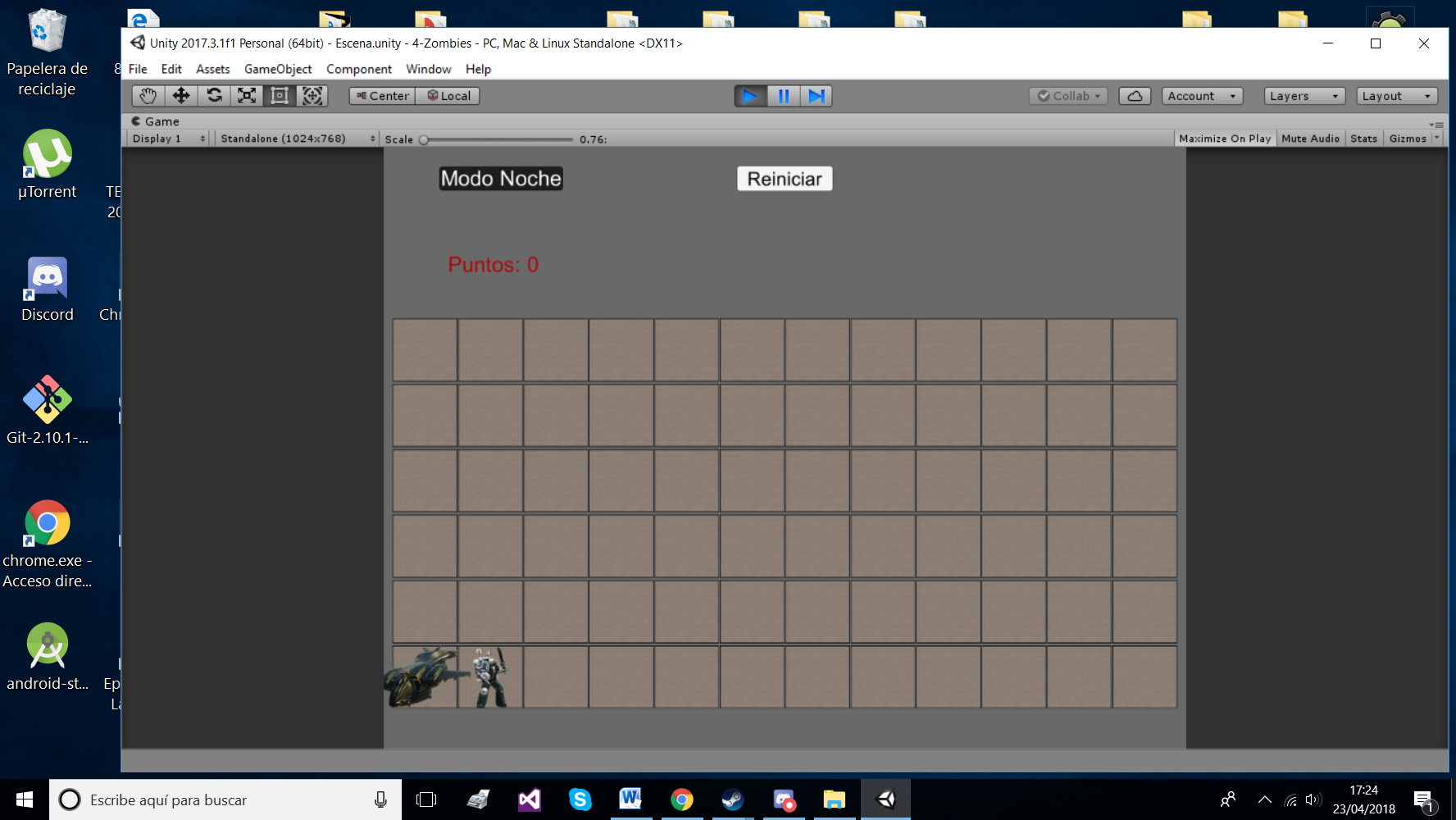
**Adicionales:**

1. Botón para reiniciar simulador en cualquier momento.
2. Sólo se puede comenzar la partida si hay al menos un zombie en el mapa.
3. Posibilidad de hacer jugar al usuario. Las decisiones del agente (Atacar, Retroceder o Esperar) las puede decidir tanto el ordenador mediante el algoritmo o el usuario según le convenga.
4. Texto que muestra los puntos y el turno actual que corresponda.

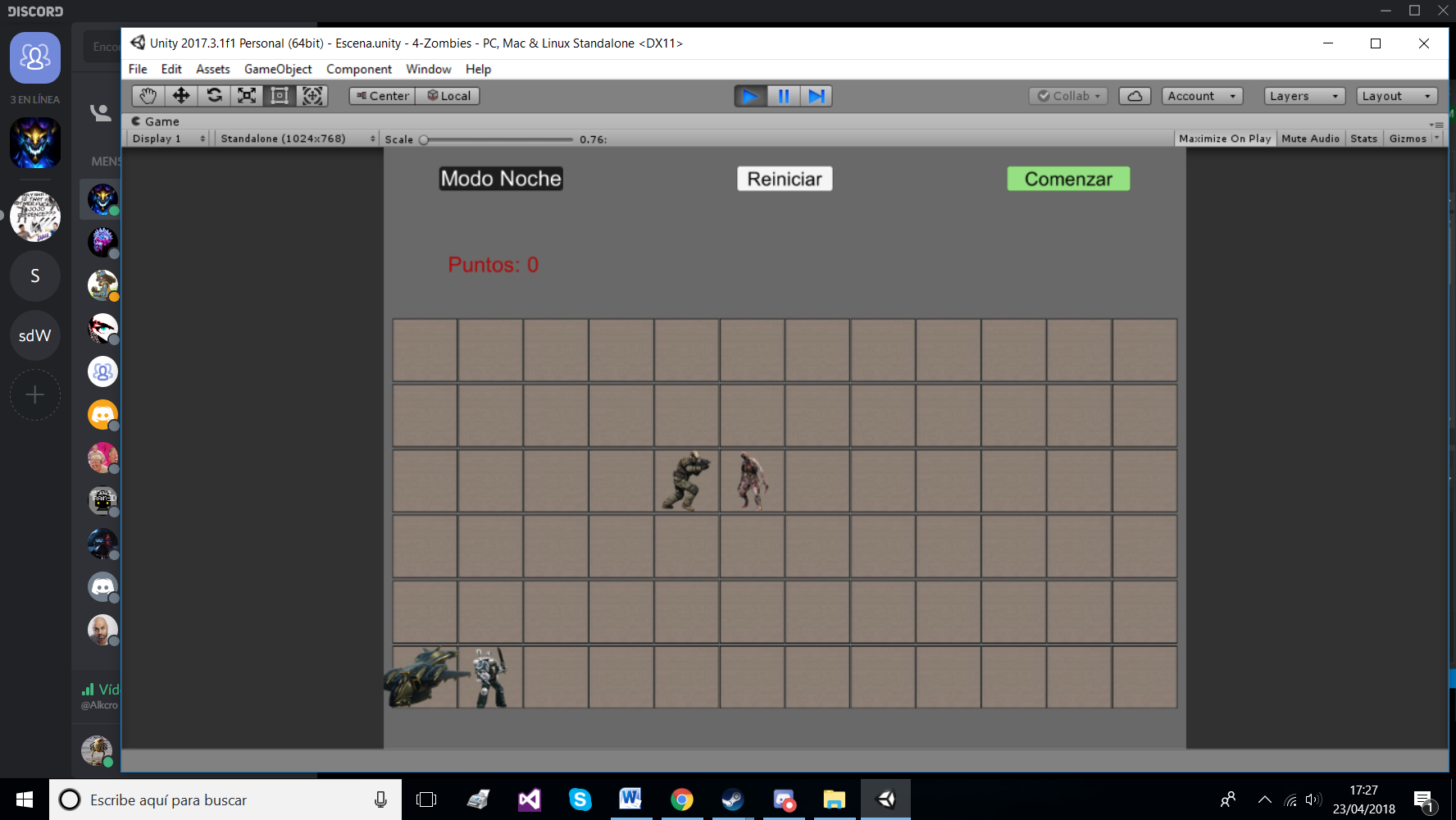
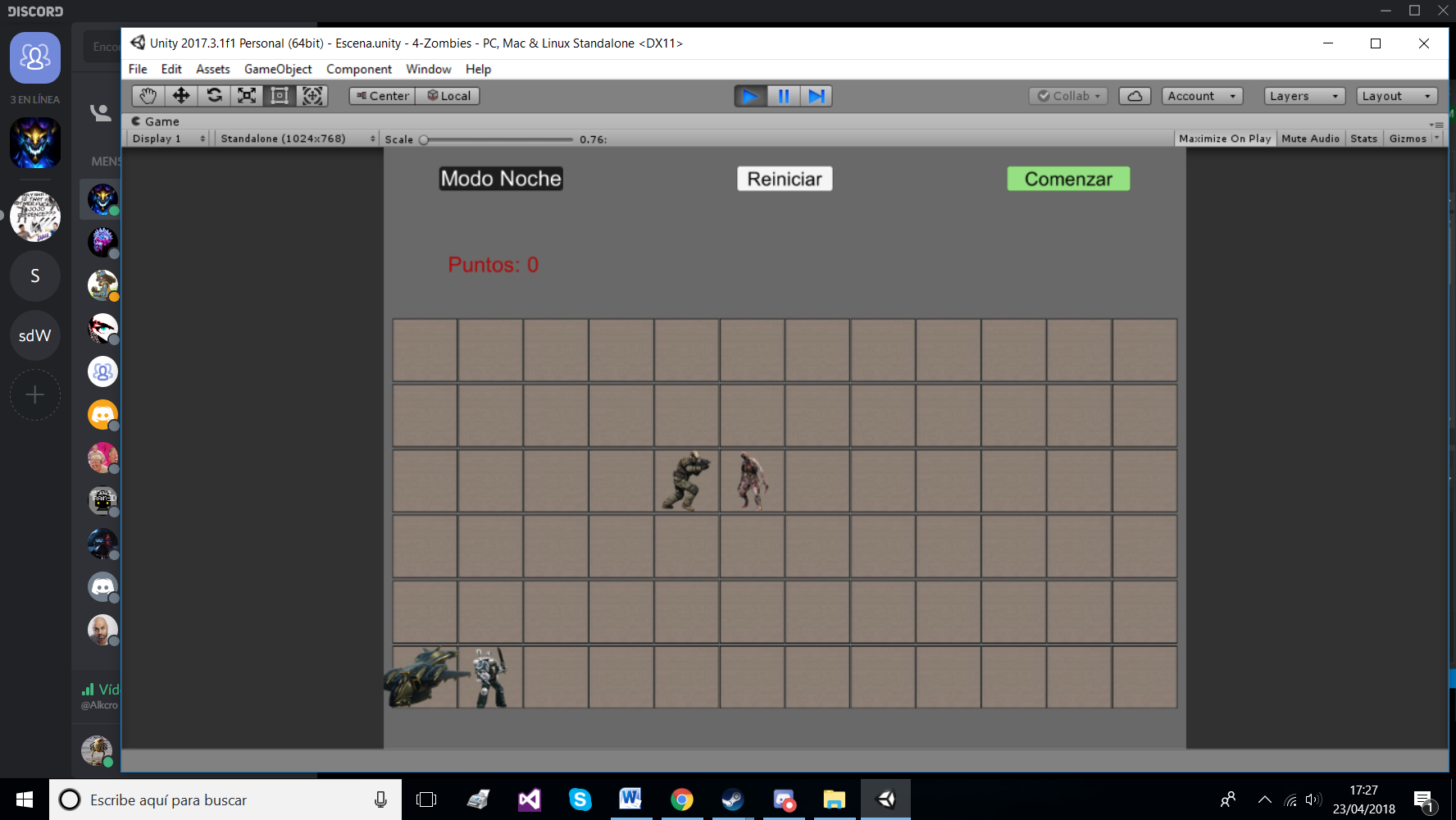
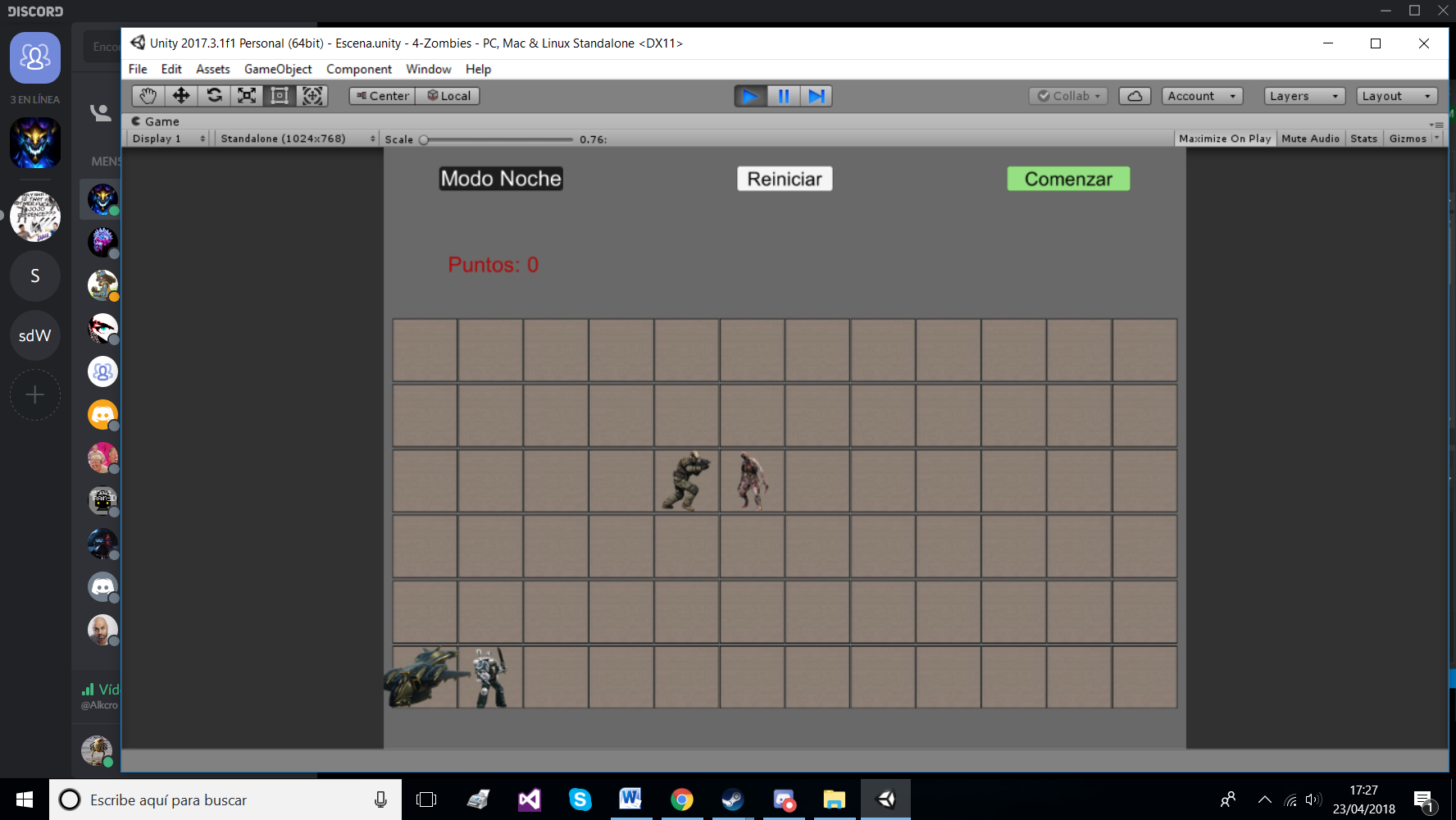
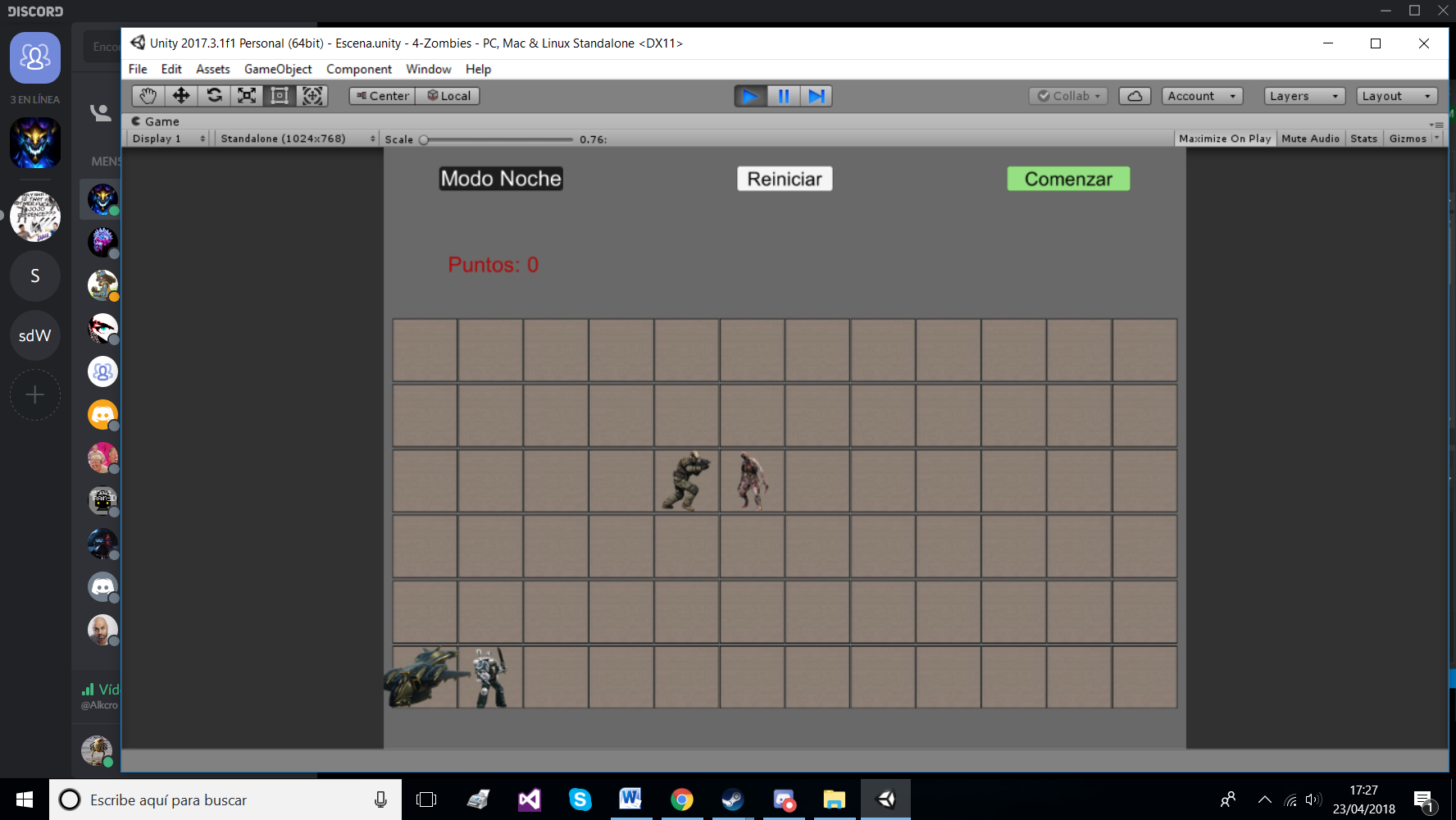
El juego

Repasamos el funcionamiento final del simulador.

Ésta es la vista nada más iniciar el juego:



Al presionar cualquier casilla (excepto las que están ocupadas por la nave y el agente) aparece un aliado y si se vuelve a hacer click en su lugar aparece un zombie.

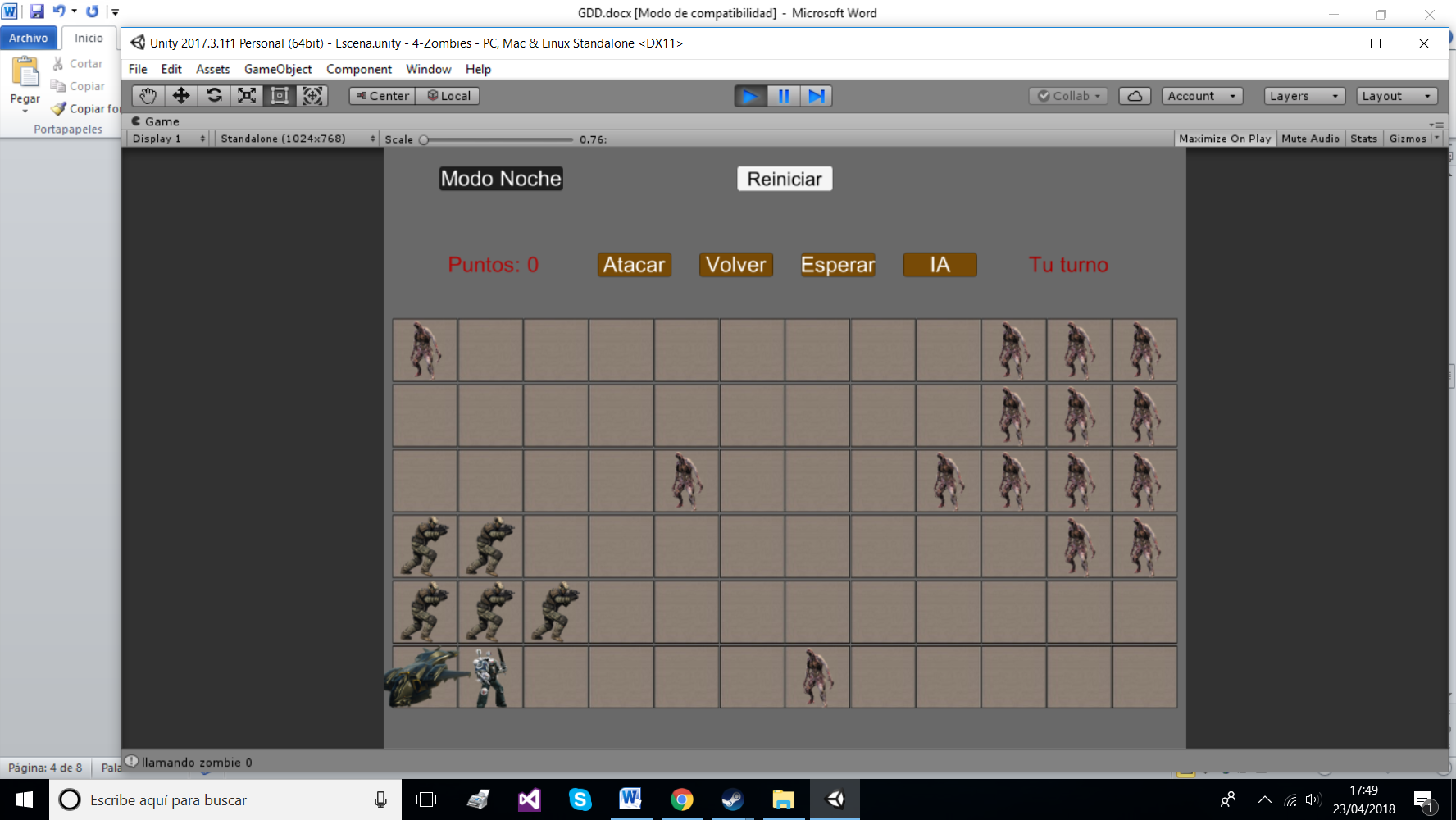


Una vez haya al menos un zombie en el mapa, se puede comenzar la batalla. La batalla se desarrolla por turnos, el turno de los zombies y el turno del agente. El primer turno se decide de forma aleatoria.

En cualquier momento de la partida se puede reiniciar o cambiar entre modo noche/día, pero no se puede clickear una casilla para añadir entidades si se ha comenzado ya la batalla.

**El turno del agente**

En el turno del agente se muestra el siguiente menú de acciones:



El usuario decide qué acción hace el agente o si lo decide la IA por él.

Tal como en el enunciado de la práctica, las acciones para cada turno son:

1. **Atacar** al zombie más cercano (avanza una casilla hacia él y hay batalla si entra en la misma casilla).
2. **Retroceder** una casilla hacia la nave.
3. **Esperar** y permanecer en el sitio.

**El turno de los enemigos**

Los zombies sólo buscan atrapar a los agentes y acabar con ellos. La única acción que realizan por lo tanto es avanzar una casilla hacia el agente más cercano o aliado, y atacar si entra en la misma casilla. Cada zombie tiene su turno independiente y espera a que haya terminado el turno del zombie anterior.

Los aliados **no** entran en los turnos, se quedan en la posición que estén para el resto de la partida.

**Los combates**

Los combates sólo tienen lugar cuando un zombie entra en la casilla de algún soldado o el héroe en la casilla de uno (o varios) zombies. En los combates se decide, mediante probabilidades, quién vive y quién muere en el conflicto.

Las probabilidades dependen de dos factores:

* La **situación**: cantidad de enemigos vs. cantidad de aliados. Es más probable que gane el héroe si dispone de más apoyo por parte de los aliados.
* La **destreza**: cantidad de aliados vs. visibilidad (si es o no modo noche).

Dependiendo de estos factores, puede suceder que muera el atacante, el atacado o ambos. Si hay varios atacados en una misma casilla se hacen los combates por separado.

**Puntuación**

Cada resultado de un combate se ve reflejado en la puntuación de la siguiente forma:

|  |  |
| --- | --- |
| **Si un aliado mata un zombie** | 1 punto |
| **Si el agente mata un zombie** | 5 puntos |
| **Si muere un aliado** | -10 puntos |
| **Si muere el héroe** | -50 puntos |

La IA del agente debe decidir la acción dependiendo tanto de los puntos como de la situación.

\*La puntuación máxima que se puede conseguir con 20 zombies es de 100 puntos. (Victoria arrasante)

\*La puntuación mínima con 5 aliados es de -100 puntos. (Fracaso total)

**Cuándo acaba la partida**

La partida acaba si muere el agente, mueren todos los zombies o se retira el agente (consigue llegar a la nave antes de que le acorralen los zombies).

Implementación

* El Script de GameManager se encarga de gestionar todo el juego antes de comenzar la partida en sí. Crea el tablero, todas las entidades (tanto las que ya están como las que se van colocando), controla el modo noche y el reinicio.
* El Script de GUI se encarga de la gestión de turnos, puntuación, eliminación de entidades por combate y cálculo de la destreza y la situación. Es decir, controla todo el GamePlay en cuanto se da a comenzar.
* Los Scripts de OnClick, como su nombre indica, realiza determinada respuesta dependiendo de qué entidad se haya clicado como se indicó más atrás (clickeado de casilla-aliado-enemigo).
* TurnoPlayer decide qué acción realizará el jugador, ya sea por IA o por usuario, y la lleva a cabo.
* TurnoEnemy controla el comportamiento de cada enemigo en el mapa, busca el objetivo más cercano y mueve una casilla hacia él cuando es su turno.

Algoritmo

**Problemas**

-Principal problema con las posiciones en el mapa al dar velocidades, al igual que en anteriores prácticas, pero ya con menos fallos.

-Problemas al principio con el array de enemigos y de aliados, donde se guardan sus respectivas entidades y, al reiniciar la partida, se salía de rango en ocasiones porque no redimensionábamos el array.

-Quisimos compactar los tres OnClick que tenemos en uno solo, pero nos daba problemas extraños ya que tratábamos de identificar la entidad que correspondía y se hacía un lío, al final fue mucho más sencillo hacerlo por separado para quitarnos de problemas.

**Resultados**

Al final hemos logrado lo siguiente:

* Que el mapa se genere tal como se pide, pudiendo colocar agujeros.
* Que se pueda cambiar al modo noche para más verosimilitud y que se pueda incrementar la velocidad.
* Que el detective encuentre el cadáver y el arma entorno a una estadística del 85-90% (5 de cada 6 veces de forma aleatoria)
* Que el detective encuentre el objetivo con la eficiencia máxima posible si no encuentra peligro de barro.
* Que el detective siempre vuelva por el camino más seguro posible y conocido a casa una vez logrado el objetivo.

NO hemos logrado los siguientes objetivos (que reúnen el 10-15% de los casos en los que el detective no encuentra el cadáver):

* Que el detective consiga salir cuando está rodeado de barro de la forma más eficiente. Además, el recorrido por las casillas descubiertas una vez encontrado barro es completamente random salvo si encuentra casillas desconocidas, que les da preferencia. De esta forma, a veces debido al random da vueltas innecesarias. Probamos con una matriz que contara el número de veces que se pasaban por las casillas para que pasara por las que tuviera menor número, pero no nos llegó a funcionar.
* El detective en ocasiones esporádicas muere cuando pasa por una casilla cercana a un agujero. Esto se debe a un fallo que por más que depuramos no conseguimos localizar, aunque sabemos lo que ocurre no sabemos el por qué. Pocas veces pero las suficientes como para fastidiarnos el simulador, el método que devuelve la posición del detective se “escapa” un frame y da una posición que no es, sólo lo hace un frame y parece como si lo hiciera al azar. Si coincide justo ese frame puede dar una casilla que no es y morir en un hueco. De la misma forma puede ocurrir que el cuchillo o el arma lo coja sin estar en su casilla. Ocurre aproximadamente en torno a un 10% de las ocasiones.

Referencias y apoyos del desarrollo de la práctica:

* *Apuntes del Campus Virtual*
* *Google Imágenes*
* [*http://buildnewgames.com/astar/*](http://buildnewgames.com/astar/) *(algoritmo de la práctica anterior)*

*Repositorio de Github:* https://github.com/alex97ortega/IA.git