**Inteligencia Artificial para Videojuegos**

***Práctica 3 : La escena del crimen***



*Grupo 07*

*Alejandro Ortega Álvarez*

*Borja Cano Álvarez*

Problema a resolver

* La propuesta es representar en Unity un escenario o mapa (mediante casillas) donde ha habido un crimen, y un agente que se encargue de descubrir el lugar del crimen y el arma mediante lógica proposicional.



* Para ello se piden los siguientes requisitos:

1. Un tablero de 5x10 (posiblemente modificable), con varios tipos de casillas: normales, agujeros, embarradas, ensangrentadas, y mezcla entre las dos anteriores.
2. Al empezar, todas las casillas son normales
3. Después se coloca un cadáver en una zona aleatoria del tablero. Las casillas colindantes se llenan de sangre y aparece el arma a distancia 2 del cadáver.
4. Después se colocan agujeros en zonas del mapa que no coincidan con la posición del cadáver ni del arma.
5. Después, el detective sale de su casa a tratar de descubrir dónde está el cadáver

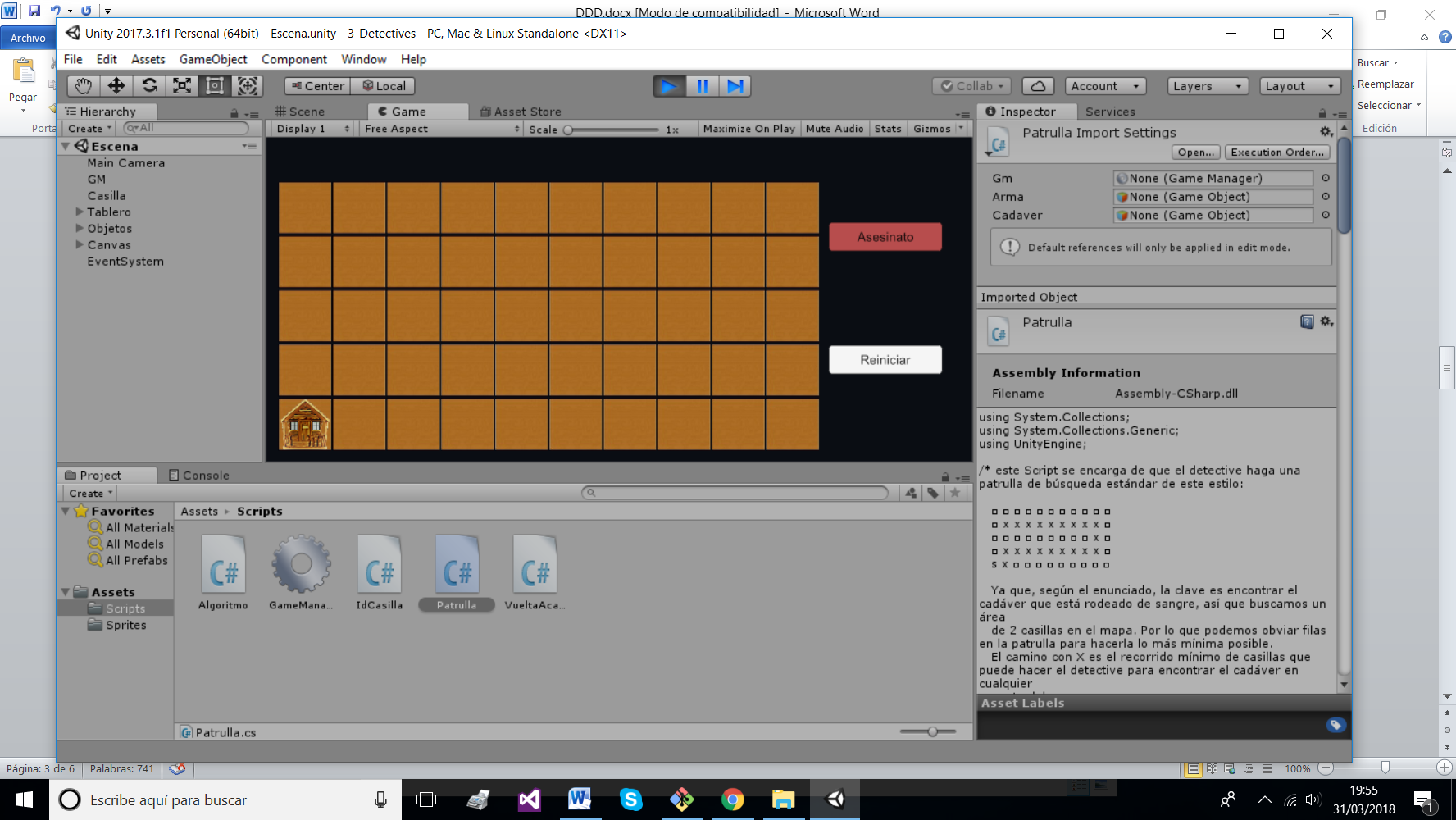
**Adicionales:**

1. Casa (lugar de donde sale el detective) en la casilla 0,0. No pueden aparecer huecos, cadáver o arma en esta casilla.
2. Botón para colocar tanto muerto como arma y agujeros (aparecen X agujeros aleatorios, donde 0 < X <= N -> valor máximo de agujeros indicado).
3. Botón para que el detective comience a patrullar, sólo si ha habido asesinato.
4. Botón para reiniciar y empezar de nuevo.
5. Botón de incrementar la velocidad de búsqueda.
6. Botón para el modo Noche.

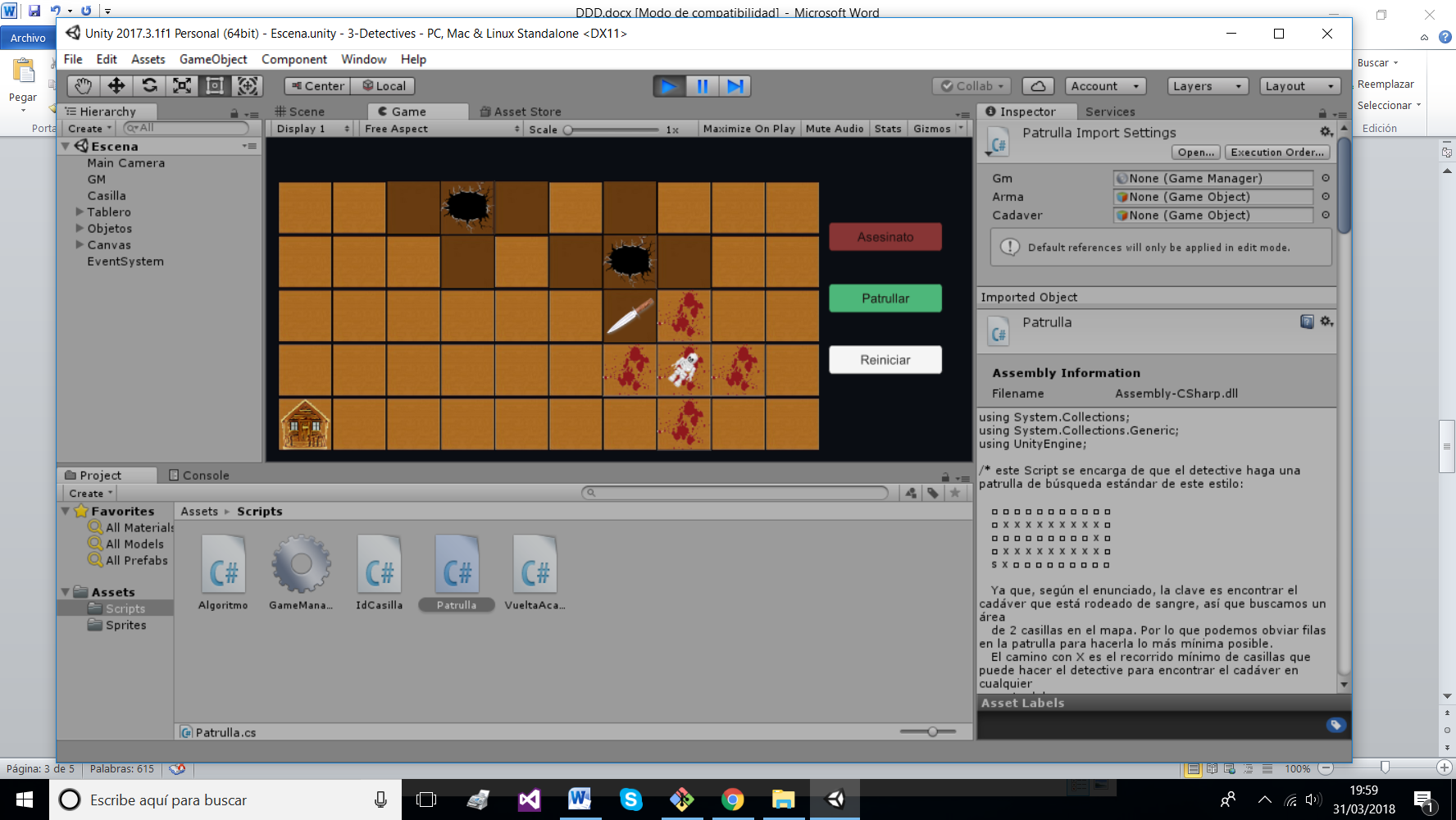
Juego final

Repasamos el funcionamiento final del simulador.

Ésta es la vista nada más iniciar el juego:

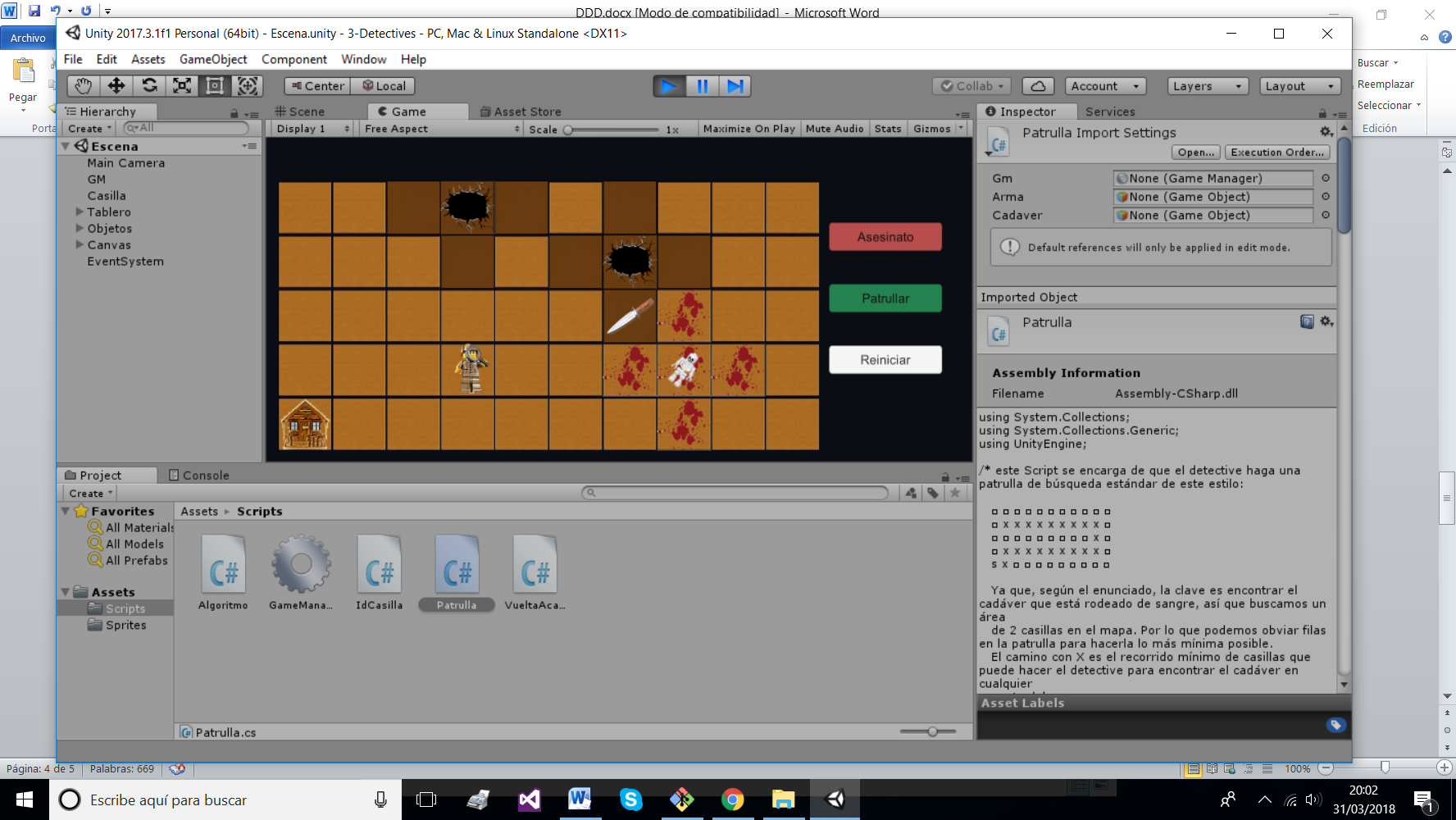
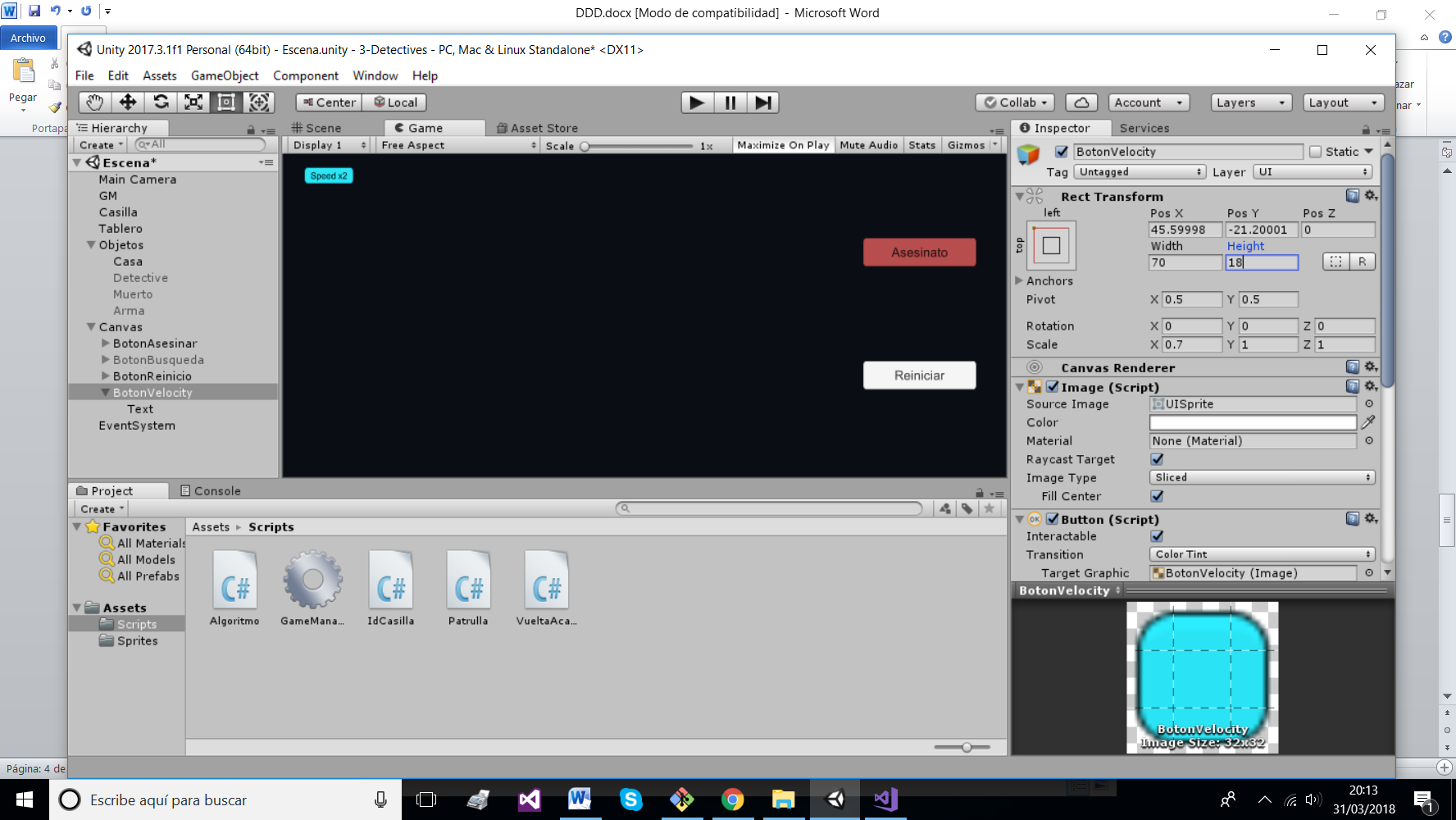


Al presionar la tecla de Asesinato, aparece el cadáver aleatoriamente y un número aleatorio de agujeros (mínimo 1, máximo el número que se ponga, está puesto 3 como predefinido). Además, si se clickea una casilla cualquiera se puede poner un hueco más para las pruebas.



Una vez hay un asesinato, aparece el botón de Patrullar para que el agente busque el cadáver. Si el agente cae a un agujero y muere, podrá patrullar de nuevo. Si el agente consigue encontrar el cadáver y el arma (que sería lo ideal), vuelve a su casa por el camino más corto.

(se puede incrementar la velocidad del agente)

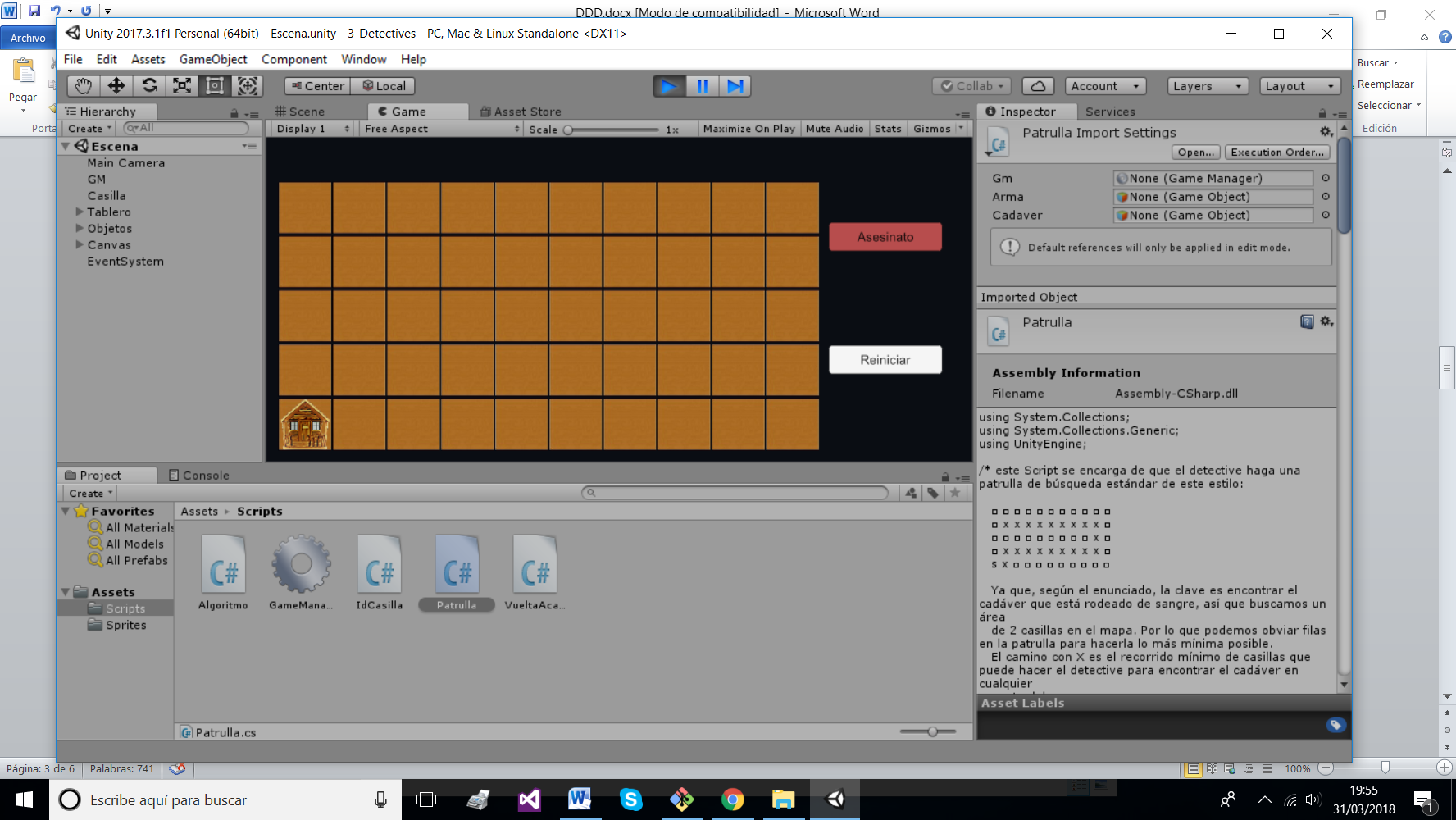


El botón de patrullar sólo vuelve a funcionar si el agente muere o hay un nuevo asesinato, ya que no tiene sentido darle una patrulla si está ya patrullando.

**La patrulla del agente**

El agente conoce bien las dimensiones de los alrededores de su casa, por oscuros y peligrosos que sean, y tiene un truco para patrullar la zona de forma eficaz y rápida, para poder volver lo antes posible a casa y no tener que pasar por casillas innecesarias o que puedan ser peligrosas.

Por ello el agente se rige a un patrón que lleva siempre a cabo cuando va a salir a patrullar. El agente siempre hace el mismo recorrido, que es el siguiente:



De esta forma, se asegura de que si ha habido algún crimen en su zona, podrá descubrirlo sea cual sea su ubicación si lo recorre de esta manera, dado que las casillas alrededor del cadáver siempre están rodeadas de sangre y por ello hay casillas que se puede saltar sin problema, si el cadáver está en alguna de ellas se asegura de que pasa por los alrededores donde estará su sangre.

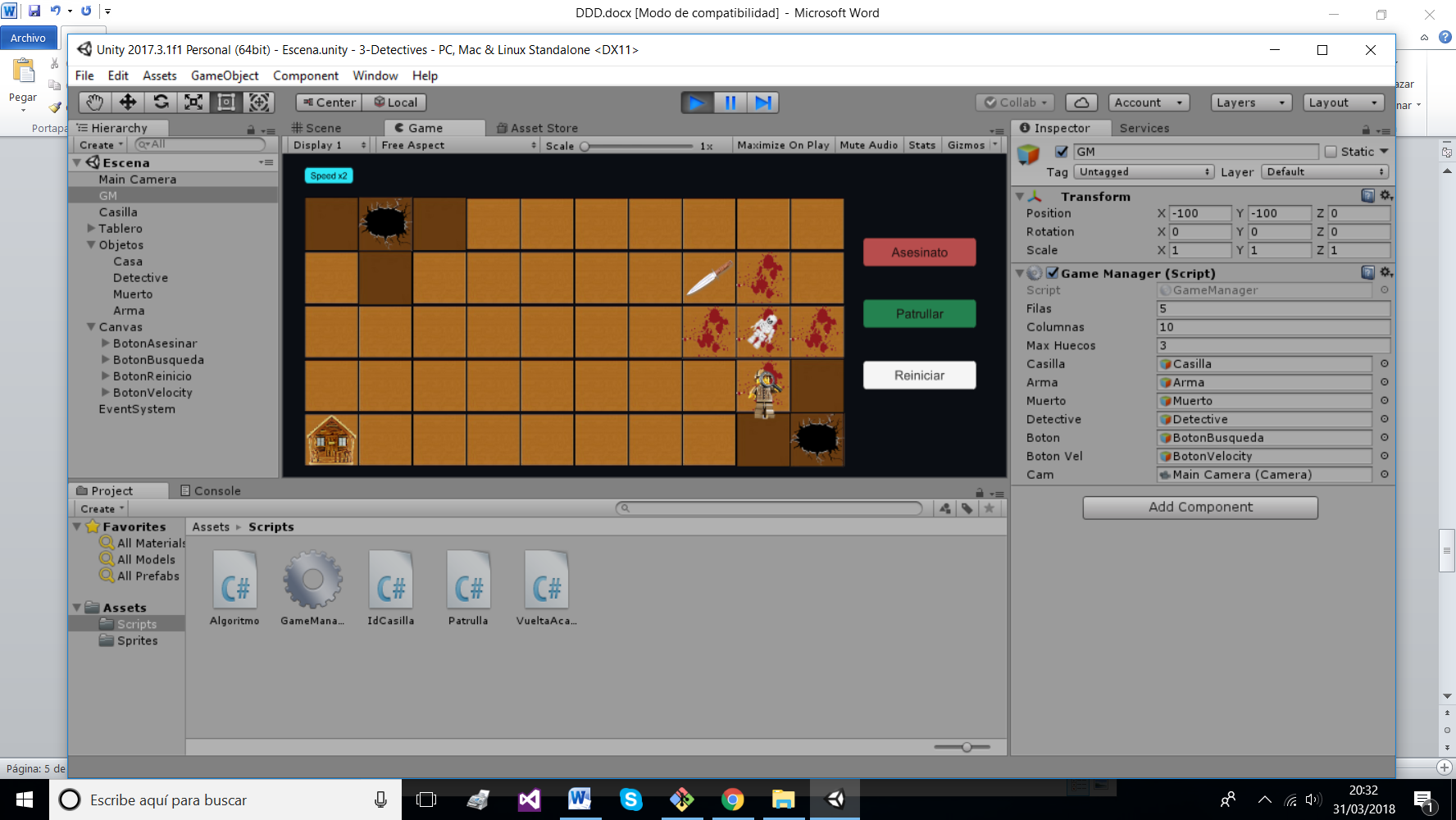
El detective sigue su patrón hasta que encuentra sangre, barro, o el arma. (nunca podrá encontrarse de primeras el cadáver o un agujero).

**Si encuentra Barro**

El detective no se la puede jugar yendo a una casilla desconocida si encuentra barro, por ello tiene que volver a una casilla segura para probar otro camino (cancela su patrón de patrulla).

**Si encuentra Sangre**

En cuanto el agente encuentra sangre, ya está más cerca de lo que busca. El agente sabe que el cadáver está en alguna de las casillas adyacentes a la sangre que acaba de encontrar, así que sólo le queda buscar en tres casillas (de la que venía no hace falta que vuelva a pasar porque ya sabe que ahí no está), y de esta forma encontrar el cadáver o también puede que el arma.



**?**

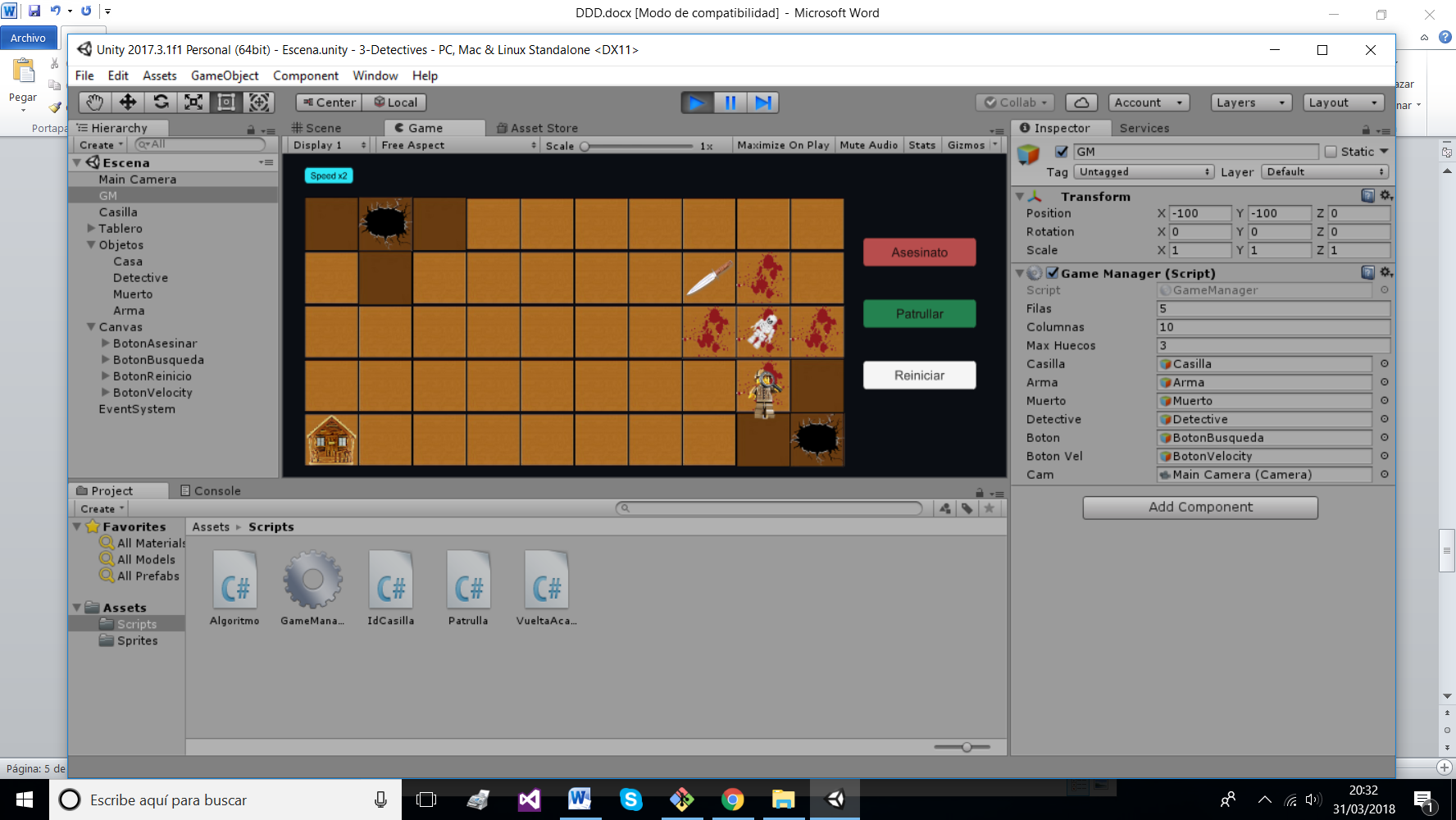
**?**

**?**

**Si encuentra el Arma**

(Se hace el mismo procedimiento si encuentra el cadáver **después** de haber tocado sangre)

El detective está aún más cerca de lo que busca, ya tiene la mitad de las piezas. Sólo hace falta que busque la otra mitad. Como el cadáver está a dos casillas como máximo del arma, eso quiere decir que puede hacer un recorrido de 3x3 para encontrarlo, sabe con certeza que tiene que estar en éste área:



**?**

**?**

**?**

**?**

**?**

**?**

**?**

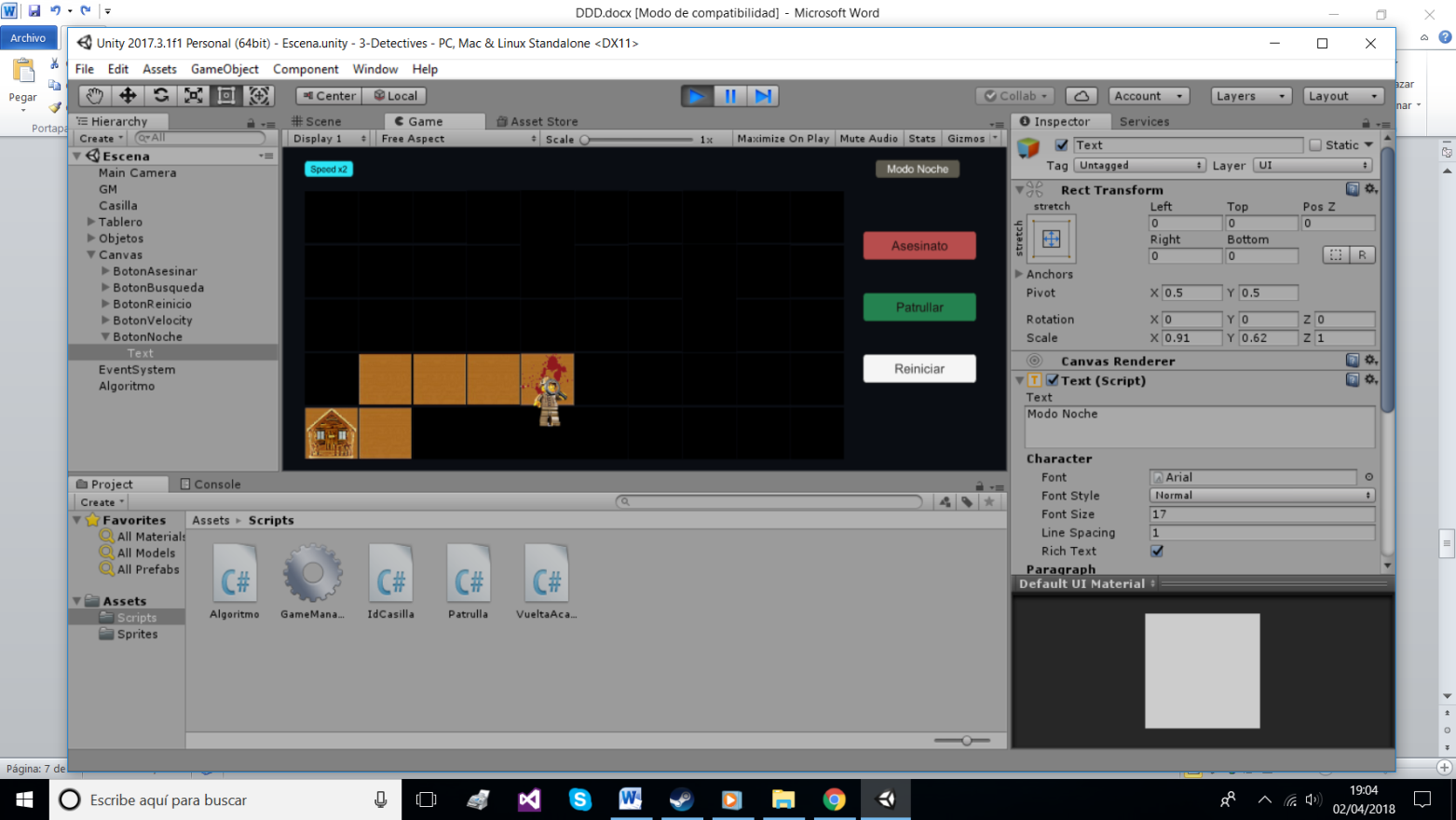
**?**

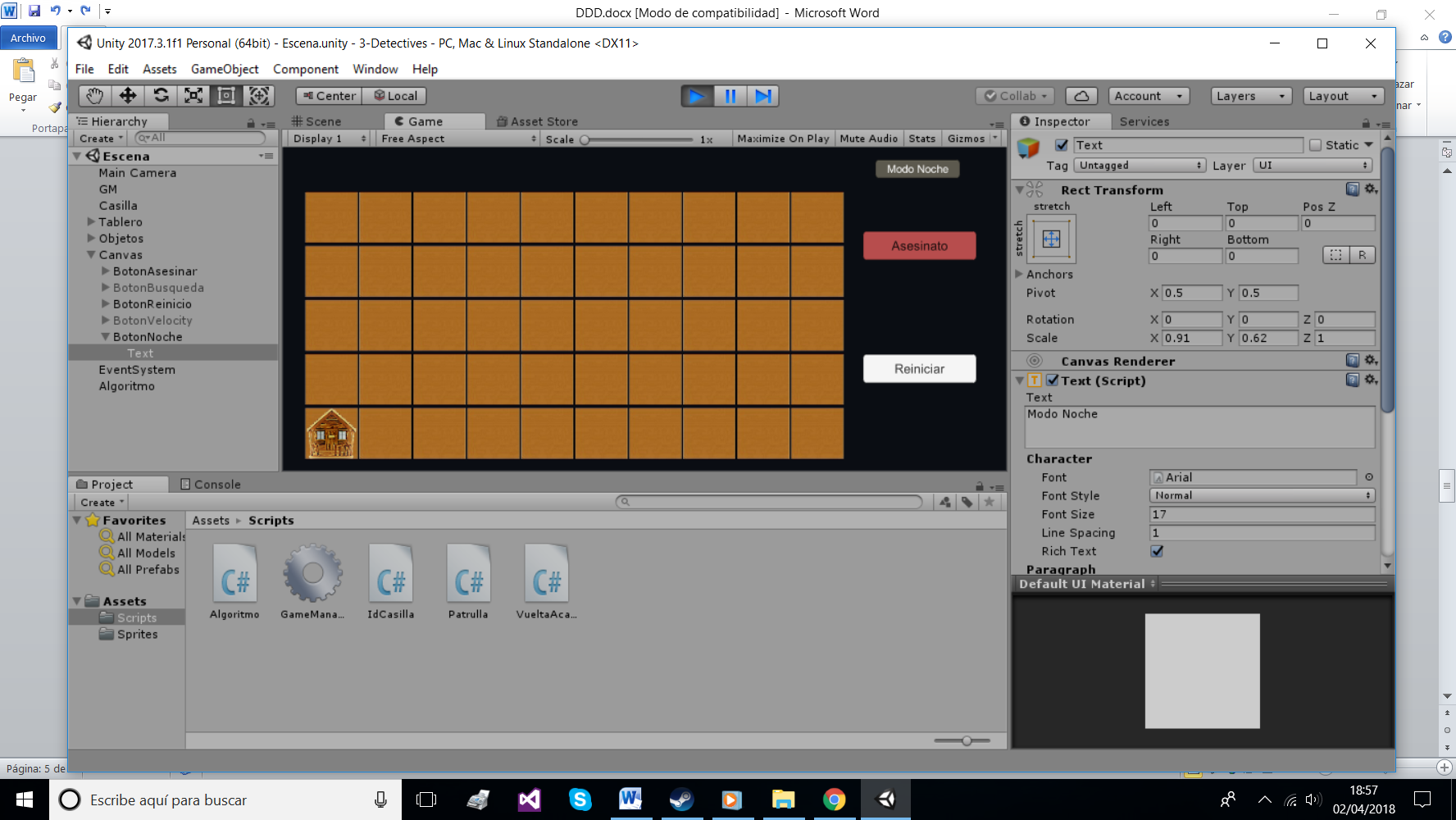
Por lo que sólo le queda investigar en esas 8 casillas. Si ha encontrado las dos cosas, arma y cadáver, vuelve a casa por el camino más corto de entre las casillas conocidas, y finaliza el caso.

**El modo noche**

El modo noche es un elemento adicional que da un poco más de verosimilitud al simulador.

Todo el mapa se oscurece y sólo se pueden ver las casillas que vaya descubriendo el detective por el mapa. En realidad, con ello estamos poniéndonos en la piel del detective ya que su comportamiento está hecho para recorrer en mapa sin saber dónde están las cosas, así que en el modo noche podemos decir que vemos sólo lo que ve el detective.





Si se vuelve a dar al botón, simplemente se “vuelve de día” y podemos ver el mapa actual, dónde está cada cosa.

Implementación

* El Script de IdCasilla se encarga de devolver el identificador que posee cada casilla, con un enumerado que hace referencia a cada tipo.
* El Script de GameManager gestiona todos los cambios de estado del juego (creación inicial del mundo, realizar un asesinato, colocar agujeros y reiniciar).
* El Script de Patrulla se encarga de gestionar los distintos comportamientos del agente antes de haber resuelto el crimen, explicados anteriormente (patrón de patrulla inicial, encontrar sangre, encontrar barro, encontrar arma y cadáver).
* El script de Algoritmo devuelve una lista con el camino más corto a casa desde la posición del detective entre las casillas que actualmente estén descubiertas
* El Script de VueltaAcasa se llama al resolver el crimen y llama al Algoritmo para que le devuelva el camino de vuelta a casa y lo procesa para darle las velocidades correspondientes.

Algoritmo

**Problemas**

-El problema más importante que nos ha ocurrido ha sido el mismo que para la práctica de los tanques: la posición exacta por el tablero del detective, dado que le aplicamos una velocidad y su posición en sí misma muchas veces no va a coincidir con la de un entero por una millonésima de unidad.

Esto nos obliga a redondear y/o truncar la posición, y pronto nos dimos cuenta que esto dependía de la dirección de la velocidad (si se va para abajo o a la derecha, hay que truncar la posición, a la inversa hay que redondearla). Con esto ya casi lo teníamos, pero nos seguían ocurriendo cosas raras y después de varios días investigando, descubrimos que sólo modificábamos la posición en X (con respecto a si es truncamiento o redondeo) si había cambios en la dirección en X, y lo mismo para la Y, por lo que dependía completamente de la suerte si en el momento de cambio de dirección en X, la Y era correcta siendo truncada o redondeada.

En resumen, había que guardar el estado de cambio de dirección anterior tanto para X como para Y, para saber si truncar o redondear cuando vale 0 la velocidad en ese eje.

Aun así, aunque ya funcionan bien las posiciones y no se sale nunca de rango, en ocasiones esporádicas, al pasar al lado de un agujero sin tocarlo o del arma, hace como que pasa por encima, y creemos que sigue siendo debido a algo de esto.

-Hubo problemas a nivel de código, al haber un montón de opciones posibles en la patrulla del agente (si encuentra sangre cambia el modo de búsqueda, lo mismo si encuentra el arma, si ya ha acabado, si encuentra barro, si ha muerto, etc.) y eran muchas líneas de código. Pero más o menos lo conseguimos abstraer todo con un enumerado que controlara el estado del agente para saber qué debía hacer, y con eso nos ahorramos unas cuantas líneas de código.

-Otro problema notable fue a la hora de decidir qué debía hacer el agente al encontrar barro. Discutimos varias opciones, que rodeara siempre en un gran área, que pasara y siguiera con la patrulla a ver si había suerte, que decidiera de forma aleatoria qué dirección tomar al llegar a una casilla de barro (era una buena opción pero muy costosa ya que requería reestructurar el sendero que habíamos puesto fijo y rompía los esquemas de la patrulla). Al final se decidió que si accedía a una casilla embarrada no se la podía jugar yendo a una casilla desconocida, por lo que rompía los esquemas de patrón de búsqueda inicial pero al menos ahora la mayoría de los mapas los puede pasar.

**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 80-85% (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 15-20% de los casos en los que el detective no encuentra el cadáver):

* Que el detective consiga salir cuando está rodeado de barro, ya que nos resultó imposible a código detectar esto, y hace bucles infinitos que no conseguimos parar. Si se ve desde fuera es muy sencillo ya que visualmente está claro por dónde debería ir, pero a código es bastante más difícil. Intentamos con una matriz de enteros que indicara el número de veces que se había pasado por una casilla, pero no resultó. Por lo que la única solución que se nos ocurrió es con un random. Aleatoriamente, el detective decide arriesgarse a una casilla desconocida si está en barro o no, pero este random tiene más posibilidades de que de que no, para que sea más común que busque otro camino. Además, aprovechamos la matriz y el random depende del número de veces que se haya accedido a la casilla. Solucionamos el problema entonces pero no de la manera más eficiente.
* 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.

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

*Falta:*

*Que haga cosas chulas cuando ve barro*

*Terminar DDD*

*Arreglar bugs (que no pase por casillas innecesarias, que no haya situaciones “imposibles”, la mierda de los huecos que se muere cuando no debe)*