2021-2-IT02-KA210-SCH-000048086
Erasmus+

PLAY

PLAY

PROTECT, LEARN, ENGAGE AND ENJOY

Toolkit

Paidea
A.M.E.F.E
asociación malagueña de
educación y formación europea

Módulo 1

Módulo 2

Módulo 1

Unidad 1 - Visión general y objetivos del proyecto

Play - proteger, aprender, comprometerse y disfrutar es un proyecto financiado por la Unión Europea en el marco del programa Erasmus+, el programa de la UE en los ámbitos de la Educación, la Formación, la Juventud y el Deporte.La asociación está formada por una escuela italiana y otra polaca, una Asociación Española de Educación y Formación Europea y Paidea, una empresa italiana de edtech. La idea surgió de la voluntad de hacer de la escuela un entorno en el que los estudiantes puedan experimentar libremente y construir sus conocimientos utilizando un enfoque práctico a través del apoyo de la tecnología. Lo que se enseña en la escuela debe estar directamente relacionado con el mundo en el que vivimos y debe potenciar el desarrollo de las competencias digitales necesarias en una sociedad tecnológica y en constante evolución, siguiendo las directrices del Digcomp.

El proyecto PLAY pretende fomentar las competencias digitales de los estudiantes de 11 a 14 años y concienciarlos sobre la importancia de proteger nuestro planeta haciéndoles participar directamente en el desarrollo de un videojuego a través de la plataforma Unity 3D y haciéndoles aprender el lenguaje de programación C#.
Uno de los Objetivos de Desarrollo Sostenible previstos por la Agenda 2030 destaca la necesidad de adoptar medidas urgentes para combatir el cambio climático y sus repercusiones. PLAY utiliza un enfoque lúdico para concienciar a los estudiantes sobre la importancia de modificar nuestro estilo de vida y potenciar el desarrollo de competencias digitales que representan un importante conjunto de habilidades en el mundo laboral. Tras una formación específica, los alumnos desarrollaron un videojuego en 3D en el que es posible interactuar con diferentes entornos para comprender las principales causas del despilfarro de recursos y la contaminación, así como conocer alternativas sostenibles que contribuyan a reducir nuestro impacto medioambiental.

Unity es un motor de juegos multiplataforma utilizado principalmente para desarrollar videojuegos y simulaciones para PC, consolas, dispositivos móviles y sitios web. Se puede programar en C#, Java, UnityScript o Boo, un lenguaje similar a Python. En el proyecto el lenguaje de programación seleccionado ha sido C# por ser muy fácil de usar, por lo que es adecuado para desarrolladores principiantes.

Han desarrollado el videojuego con la supervisión de expertos programadores y formadores. Más concretamente, la actividad ha previsto las siguientes fases:

- Formación y práctica. En la primera fase, los alumnos han recibido formación sobre cómo utilizar Unity y cómo programar en C# (junto con los profesores); tras recibir una base teórica, han practicado cómo escribir código eficiente.
- Plan de desarrollo. Se han organizado varias sesiones de brainstorming, en las que los alumnos han podido debatir sus ideas y cooperar. Después, han tenido que investigar sobre las principales causas del cambio climático y sobre soluciones más ecológicas. Esto llevó a la selección de seis escenarios, que representan el fondo interactivo y los niveles del juego.
- Producción. Es la fase de desarrollo real del juego.
- Prueba y validación. Una vez terminada la producción, la versión final ha sido probada por los alumnos de los centros asociados que no han participado en el proyecto.

Los profesores han participado activamente en todas las fases de producción de esta actividad, lo que les ha permitido aprender métodos pedagógicos innovadores y también desarrollar nuevas competencias.

Los principales objetivos del proyecto pueden resumirse como sigue:

1. fomentar en los jóvenes estudiantes el desarrollo de competencias digitales y de programación que les hagan competitivos en el mercado laboral;
2. sensibilizar a los estudiantes y a la comunidad en general sobre las causas del calentamiento global y sobre la importancia de luchar contra el cambio climático modificando nuestros hábitos;
3. promover el desarrollo de métodos de enseñanza innovadores y digitales, dotando a los profesores de los conocimientos y competencias necesarios para aumentar el nivel de motivación de los alumnos y su rendimiento académico.

Al desarrollar el videojuego, los alumnos han tenido la oportunidad de experimentar de forma práctica con la programación y han adquirido competencias digitales que les serán útiles en su futura vida profesional. La programación informática desempeña un papel integral en nuestro mundo. Aprender los fundamentos de la programación puede dar a los estudiantes una ventaja competitiva en este mundo impulsado por la tecnología. Los conocimientos de programación también son importantes para aprender a innovar y crear soluciones ecológicas para problemas globales. Al mismo tiempo, los alumnos han ampliado sus conocimientos sobre un tema tan crucial y se han convertido en personas responsables que asumirán un papel activo en la lucha contra el cambio climático. Tomar conciencia de tal amenaza y ayudar a los demás a comprender las consecuencias de nuestras acciones ha tenido un impacto importante en la vida de los participantes, haciéndoles más responsables y conscientes del medio ambiente y de su papel en la sociedad. Las actividades previstas también pretendían fomentar la inclusión y el trabajo en equipo. Por supuesto, algunos alumnos tenían más conocimientos digitales y otros estaban más interesados en el medio ambiente y sensibilizados con este tema. Trabajando juntos tuvieron la posibilidad de aprender unos de otros.

En los nuevos programas de aprendizaje es necesario utilizar metodologías adecuadas, por ejemplo la gamificación o el aprendizaje basado en juegos, para implicar a los estudiantes, estimular su creatividad, adaptar los programas de aprendizaje y hacer más atractivos los contenidos. Cada uno de estos aspectos representa una variable

clave que los profesores deben tener en cuenta para alcanzar los objetivos pedagógicos y de aprendizaje. Esto exige que los profesores posean una importante Competencia Digital Docente: esto capacita al profesor para el uso de

tecnología no sólo como apoyo a sus prácticas actuales, sino también para utilizarla en la reconstrucción de los entornos educativos y de formación.

Unidad 2 - Marco de competencias digitales para educadores (DigCompEdu)

El Marco Europeo para la Competencia Digital de los Educadores (DigCompEdu) es un marco científicamente sólido que describe lo que significa para los educadores ser digitalmente competentes. Proporciona un marco de referencia general para apoyar el desarrollo de competencias digitales específicas de los educadores en Europa. DigCompEdu está dirigido a educadores de todos los niveles educativos, desde la educación infantil hasta la educación superior y de adultos, pasando por la educación y formación general y profesional, la educación para necesidades especiales y los contextos de aprendizaje no formal.

Las competencias digitales son necesarias en el aprendizaje y en el trabajo y nos convierten en miembros activos de la sociedad. En relación con la educación, tan importante como comprender las competencias en sí es saber cómo desarrollarlas.

¿Qué son las competencias digitales?

La competencia digital es uno de los elementos de las ocho competencias clave y se refiere al uso consciente y crítico de todo el espectro de tecnologías digitales que permiten la adquisición de información, la comunicación y la resolución de problemas básicos en todos los aspectos de la vida.

¿Qué es el marco DigCompEdu?

El marco distingue 6 zonas de cobertura:

1. Compromiso profesional: uso de tecnologías de comunicación eficaces, cooperación y desarrollo de la comunicación.
2. Recursos digitales: uso y aprovechamiento de recursos, creación, evaluación y puesta en común de recursos.
3. Enseñanza y aprendizaje: utilización de distintos métodos y herramientas en la enseñanza y el aprendizaje.
4. Evaluación - control de diversos métodos y herramientas de examen y retroalimentación.
5. Apoyar el aprendizaje: utilizar las tecnologías de fabricación para potenciar y hacer más accesibles, exclusivos, inclusivos y centrados en el aprendizaje los

entornos de aprendizaje.

6. Apoyar las competencias de los alumnos: apoyar y facilitar el desarrollo de sus competencias.

El Marco Europeo de Competencias Digitales de los Profesores es un marco normalizado que proporciona una referencia y una guía para trabajar con las competencias digitales de los profesores. Identifica 6 categorías útiles de actividades de aprendizaje digital que se desglosan en competencias específicas.

El marco pretende mostrar cómo pueden utilizarse las tecnologías digitales para apoyar el desarrollo profesional en la educación y la formación. No se hace hincapié únicamente en las competencias técnicas, sino sobre todo en el contexto de la enseñanza y el aprendizaje. El propio marco pretende "proporcionar un lenguaje y una lógica comunes como punto de partida para desarrollar, comparar y debatir diferentes instrumentos para el desarrollo de la competencia digital del profesorado a nivel nacional, regional o local".


Educators' professional competences
Educators' pedagogic competences
Learners' competences
Digital competences
Transversal competences
Subject specific competences
Subject specific competences
1 Professional Engagement
2 Digital Resources
3 Teaching and Learning
4 Assessment
5 Empowering Learners
6 Facilitating Learners' Digital Competence

Para comprender mejor la naturaleza de las competencias digitales, la Comisión Europea ha elaborado el Marco Europeo de Competencias Digitales para los Ciudadanos (DigComp), que se ha dividido en cinco áreas: información y análisis de datos; comunicación y cooperación; creación de contenidos digitales; seguridad; y resolución de problemas. En total, abarcan 21 competencias.

El desarrollo continuo de las competencias digitales concierne tanto a los niños como a los profesores. Todo el mundo, independientemente de su edad, si adquiere las competencias de las nuevas tecnologías, facilitará su funcionamiento en la sociedad moderna.

Es necesario formar a los niños sobre su responsabilidad en la aplicación y preparación para la vida en el mundo moderno, donde el mensaje principal es que los medios de comunicación y las tecnologías se están desarrollando rápidamente. También es necesario mejorar la formación de los profesores en el campo de la enseñanza con el uso de la tecnología moderna de tal manera que apoyen a propósito un proceso didáctico dirigido, apoyando el aumento de la eficacia de la enseñanza y el aprendizaje. Las recomendaciones del Parlamento Europeo y del Consejo y muchos actos jurídicos en materia de educación demuestran que las actividades educativas se centran en el desarrollo de los científicos que las estudian para la vida en la sociedad moderna. Una de estas competencias es la competencia digital.

Para más información, visite la siguiente página web: https://joint-research-centre.ec.europa.eu/digcompedu_en

Unidad 3 - Gamificación

Hay muchas definiciones de **gamificación**. Según Brian Burke (vicepresidente de investigación de la empresa tecnológica Gartner y experto en arquitectura empresarial) "La gamificación es el uso de la mecánica de juego y el diseño experimentado para involucrar y motivar digitalmente a los jugadores para que alcancen sus objetivos". El diccionario de Oxford define la gamificación como "la aplicación de elementos típicos del juego (por ejemplo, puntuación, competición con otros, reglas de juego) a otras áreas de actividad,[...] para fomentar el compromiso". Según el diccionario de Cambridge, la gamificación es "la práctica de hacer que las actividades se parezcan más a los juegos para hacerlas más interesantes o divertidas". Según Yu-kai Chou (autor y conferenciante internacional sobre gamificación y diseño conductual), hay ocho factores fundamentales que hacen que los juegos sean atractivos para las personas:

1. **Epic meaning & Calling** → Formas parte de algo más grande que tú mismo.

2. **Desarrollo y logro** → Estás motivado porque sientes que mejoras y alcanzas la maestría.

3. Empoderamiento de la creatividad y Feedback → Dar a la gente incluso algunos elementos simples como bloques de construcción LEGO® a la gente y dejarles libres para encontrar su propia manera de combinarlos es un proceso muy atractivo. Como hicieron nuestros alumnos cuando se les pidió que crearan unas "marionetas" utilizando material de desecho de plástico y papel (*fotos*).

4. **Propiedad y posesión** → Cuando sientes que algo te pertenece, quieres mejorarlo, quieres protegerlo y quieres conseguir más.

5. **Influencia social y parentesco** → Nos influye lo que hace la gente que nos rodea.

6. **Escasez e Impaciencia** → Se relaciona con el hecho de que a veces quieres algo solo porque no puedes tenerlo.

7. **Imprevisibilidad y curiosidad** → Si no sabes qué va a pasar a continuación, siempre estás pensando en ello.

8. **Pérdida y evitación** → Si haces algo quieres evitar la pérdida, porque ninguno quiere que ocurran cosas malas.

En opinión de Yu-kai Chou, todo lo que hacemos se basa en uno o varios de estos impulsos fundamentales. Según otros estudios, los juegos hacen a los niños más inteligentes, porque los videojuegos presentan fundamentalmente un proceso continuo de aprendizaje a los usuarios. Evolucionan y avanzan constantemente. Los juegos están diseñados para producir una reacción determinada en las personas. Un juego representa un reto y cuando lo superas se libera dopamina en tu cerebro, así que tenemos una mejora del aprendizaje, la multitarea aumenta la conexión y un fuerte bucle de dopamina en el cerebro y esto produce un refuerzo intrínseco.

Desde 2010 la Gamificación se ha convertido en una metodología en la educación en todo el mundo. Los estudios demuestran que transformar los objetivos educativos a través de retos emocionantes, utilizando modelos de videojuegos, por ejemplo utilizando insignias de progreso o dando la posibilidad de ver gráficos de rendimiento, satisface la necesidad de las personas de ser competentes en una actividad específica

y aumenta el valor percibido de la tarea. Gamificación no significa convertir la lección en un juego, sino introducir el elemento de los videojuegos en la educación. Los videojuegos pueden estimular la motivación, el interés, la creatividad, el sentido de pertenencia y la felicidad de las personas. Todos estos sentimientos son recursos que podemos emplear en nuestras actividades diarias.

Así que si introducimos las reglas y esquemas de los videojuegos en todas las tareas que tenemos que hacer en nuestra vida diaria podríamos completarlas con más motivación y satisfacción, obteniendo mejores resultados.

Unidad 4 - El valor educativo de utilizar un enfoque centrado en la tecnología.

La integración de los recursos de las tecnologías de la información en el aprendizaje tiene numerosas ventajas: en particular, proporciona formas alternativas de aprender, permite adquirir habilidades cognitivas y conocimientos importantes para el aprendizaje permanente y, sobre todo, fomenta la motivación. Según la Sociedad Internacional para la Tecnología en la Educación (ISTE), muchos de los puestos de trabajo de alta demanda actuales se crearon en la última década. A medida que los avances tecnológicos impulsan la globalización y la transformación digital, los profesores pueden ayudar a los alumnos a adquirir las competencias necesarias para triunfar en las profesiones del futuro. La pandemia de COVID-19 está demostrando rápidamente por qué la educación en línea debe ser una parte vital de la enseñanza y el aprendizaje. Al integrar la tecnología en los planes de estudio existentes, en lugar de utilizarla únicamente como herramienta de gestión de crisis, los profesores pueden crear experiencias de aprendizaje inmersivas y atractivas. El uso eficaz de herramientas digitales de aprendizaje en las aulas puede aumentar el compromiso de los alumnos, ayudar a los profesores a mejorar sus planes de clase y facilitar el aprendizaje personalizado. La tecnología proporciona a los alumnos información de fácil acceso, aprendizaje acelerado y oportunidades divertidas de practicar lo que aprenden. Les permite explorar nuevas materias y profundizar en conceptos difíciles, sobre todo en las disciplinas STEM.

Las principales ventajas del uso de la tecnología en la educación pueden resumirse como sigue:

- Mayor colaboración y comunicación. La tecnología educativa puede fomentar la colaboración. No sólo los profesores pueden interactuar con los alumnos durante las clases, sino que los alumnos también pueden comunicarse entre sí. A través de las lecciones en línea y los juegos de aprendizaje, los estudiantes pueden trabajar juntos para resolver problemas. En las actividades colaborativas, los estudiantes pueden compartir sus pensamientos e ideas y apoyarse mutuamente.
- Oportunidades de aprendizaje personalizado. La tecnología permite el acceso permanente a los recursos educativos. Puede utilizarse para adaptar los planes de aprendizaje a cada alumno. Los profesores pueden crear lecciones basadas en los intereses y puntos fuertes de los alumnos. Una ventaja añadida es que los alumnos pueden aprender a su propio ritmo y repasar el material para comprender mejor los conceptos esenciales.
- Curiosidad impulsada por contenidos atractivos. A través de contenidos atractivos y educativos, los profesores pueden despertar la curiosidad de los alumnos, que, según los estudios, está relacionada con el éxito académico. La creación de contenidos atractivos puede implicar el uso de RA, vídeos, podcasts y, por supuesto, juegos.

La introducción de elementos tecnológicos y videojuegos en los procesos de aprendizaje en entornos educativos está cada vez más presente en los debates sobre cómo mejorar la experiencia de nuestros jóvenes en un contexto en el que lo digital se está incorporando a más ámbitos de nuestras vidas. Los educadores llevan utilizando juegos para enseñar de este modo al menos desde 1971, cuando tres estudiantes de magisterio crearon el clásico juego educativo The Oregon Trail para utilizarlo en un curso de historia de Estados Unidos. Los primeros argumentos a favor de este enfoque solían centrarse en la idea de que "los videojuegos son divertidos y, como tales, ofrecen una perspectiva intrigante para coaccionar a algunos niños a aprender" (Silvern 10). El hecho de que los juegos sean eficaces para motivar a los jugadores ha seguido siendo una razón fundamental para

enseñar con juegos alineados con el contenido. Prensky describe a los alumnos que han alcanzado la mayoría de edad en la era de Internet como "nativos digitales" que aprenden mejor de forma natural cuando utilizan la tecnología (Prensky, "Listen to the Natives" 9-13). Los estudios respaldan la idea de que los jóvenes del siglo XXI tienen una gran afinidad por la tecnología, sobre todo por los videojuegos y las redes de comunicación (Jones et al 6, Lenhart et al 2-3). Otros autores atribuyen el atractivo de los juegos a su atractiva mezcla de factores, como el desafío, la fantasía y la interactividad (Malone 162; Owston 978). Los videojuegos pueden ser una herramienta de aprendizaje muy estimulante para los jóvenes en las escuelas. Además de abordar determinados contenidos curriculares, pueden apoyar el desarrollo de ciertas habilidades

necesarias para promover buenos hábitos. Este era exactamente el objetivo del proyecto: encontrar una forma eficaz de concienciar a los jóvenes estudiantes de la necesidad de preservar nuestro planeta, implicándoles directamente y haciéndoles desarrollar habilidades técnicas y blandas.

Unidad 5 - Resultados del proyecto

Módulo 2

Unidad 1 - Plataforma Unity 3D: usos y potencialidades

INTRODUCCIÓN.

El avance de la tecnología es asombroso. Quién podía imaginar hace 100 años lo que somos capaces de hacer hoy con el apoyo de las herramientas tecnológicas.

Una de estas herramientas es Unity 3D, que nos da la posibilidad de crear espacios del mundo real. Nos gustaría destacar que, tras leer este conjunto de herramientas, los usuarios finales podrán construir su propio recorrido virtual por el escenario que deseen, siempre y cuando dejen volar su imaginación y den así su propio toque artístico al proyecto deseado.

El proyecto que se presenta en este kit de herramientas fue diseñado para personas que no tienen ningún conocimiento en Unity 3D por lo que, si quieres aprenderlo desde cero estas justo en el lugar indicado, se partirá de la configuración básica del entorno de desarrollo en Unity 3D, que es cada botón y cada menú de la aplicación, asegurando el conocimiento previo de estas utilidades, antes de comenzar a desarrollar el recorrido virtual. Se hará claridad en el manejo de las dimensiones en Unity 3D, pues como su propio nombre lo indica, maneja las 3 dimensiones (x, y, z) y cuando no se tiene práctica en el manejo del programa puede ser algo tedioso, pues es cuestión de costumbre aprender a desenvolverse con estas dimensiones. Teniendo conocimientos previos de la herramienta, comenzaremos con la creación de nuestro proyecto y si aún no tenemos claros algunos botones, las dudas se irán aclarando a medida que avancemos, y a medida que practiquemos y ganemos experiencia en el manejo de Unity 3D. Una vez finalizada la creación del primer espacio o recorrido virtual, los alumnos podrán crear su propio entorno. Uno no se imagina lo potente que es esta herramienta y lo fácil que es utilizarla, y mucho menos lo sencillo que es crear un recorrido virtual con Unity 3D, por lo que te invitamos a desarrollar este recorrido paso a paso para que puedas alcanzar los objetivos deseados.

UNITY 3D.

Unity 3D es una herramienta para desarrollar principalmente videojuegos, pocos saben utilizarla y no es necesario ser un experto en programación para poder utilizarla, ya que con unas pequeñas nociones del lenguaje requerido, se puede estructurar cualquier juego interactivo. "Unity 3D, un motor gráfico 3D para PC y Mac, se utiliza para

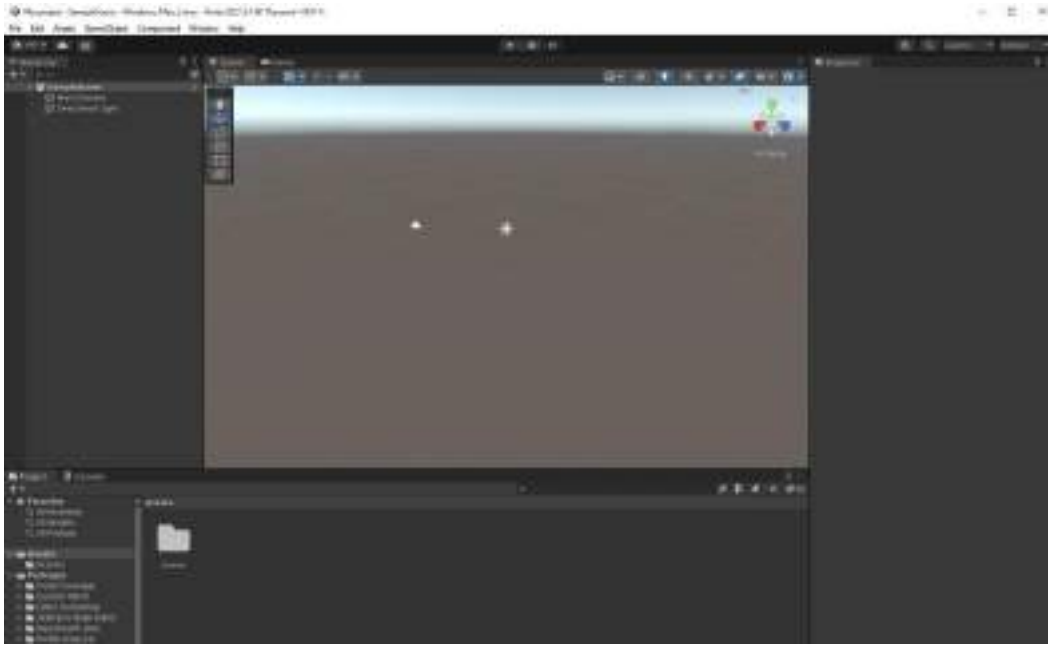
desarrollar juegos, aplicaciones interactivas, visualizaciones y animaciones 3D. Unity es compatible con plataformas como PC, Mac, Nintendo y Wii, Iphone, Android y la web utilizando su plugin "Unity web player". Unity es una aplicación creada por Unity Technologies, que "fue fundada en 2004 por David Helgason (CEO), Nicholas Francis (CCO) y Joachim Ante (CTO) en Copenhague, Dinamarca, después de que su primer juego, GooBall, no tuviera éxito. Los tres reconocieron el valor del motor y las herramientas de desarrollo y se propusieron crear un motor que todo el mundo pudiera utilizar a un precio asequible. Unity Technologies ha recibido financiación de Sequoia Capital, Capital WestSummit e iGlobe Partners. El éxito de Unity se debe en parte a que se centra en las necesidades de los desarrolladores independientes que no pueden crear su propio motor de juego ni disponen de las herramientas o licencias necesarias para aprovechar al máximo las opciones que se ponen a su alcance. El objetivo de la empresa es "democratizar el desarrollo de juegos" y hacer que el desarrollo de contenidos interactivos en 2D y 3D sea lo más accesible posible para el mayor número de personas posible en todo el mundo". Con el auge del iPhone en 2008, Unity fue uno de los primeros motores en empezar a dar soporte a esta plataforma y, hoy en día, Unity está siendo utilizado por el 53,1% de los desarrolladores, según una encuesta realizada por Game Developer sobre tecnología móvil y social, para crear cientos de juegos para dispositivos Android e iOS. Unity cuenta con un editor visual para poder crear los juegos en él, ya que todo el contenido del juego se construye a partir de este editor y la forma en que se comportan los objetos se programan utilizando un lenguaje de script (JavaScrip); lo anterior nos da a entender que no se necesita ser un experto en lenguajes como C++ para poder desarrollar un juego o una animación con Unity 3D. Unity se estructura manejando y creando escenas para el desarrollo de la aplicación deseada, una escena puede ser cualquier parte del juego o animación, ya sea un nivel del juego o un área determinada. Se empieza con un espacio en blanco en el que se puede dar forma a lo que se quiera crear utilizando las herramientas de unity. Este motor unity también incluye un editor de terreno, en el que puedes esculpir la forma del terreno utilizando las herramientas visuales que ofrece unity, puedes pintar, texturizar, añadir hierba, colocar árboles o similares, o incluso importar otros materiales de otros motores de desarrollo. Unity es accesible a cualquier tipo de público, ya que está desarrollado en varias versiones, gratuita y profesional, ambas

tienen grandes ventajas a la hora de desarrollar lo que se requiera, sin embargo, la versión más completa es la profesional, pero cabe aclarar que esta la versión gratuita tiene un costo que no todos pueden pagar y si eres alguien que apenas se está familiarizando con la herramienta, la versión gratuita es más que suficiente por el momento. Además de Unity 3D, existen otras herramientas en el mercado, que incluso pueden ser más famosas, como UDK, Epic Games o CryEngine. Sin embargo, Unity 3D tiene una gran ventaja sobre estas y es que no estamos obligados a desarrollar en sistemas operativos Windows, ya que Unity 3D también tiene una versión para sistemas operativos Mac. Unity también proporciona la facilidad de no sólo para importar terrenos, sino también modelos 3D, texturas, sonidos, etc. Con sólo unos clics, que se pueden utilizar en cualquier momento durante el desarrollo.

Unidad 2 - Interfaz

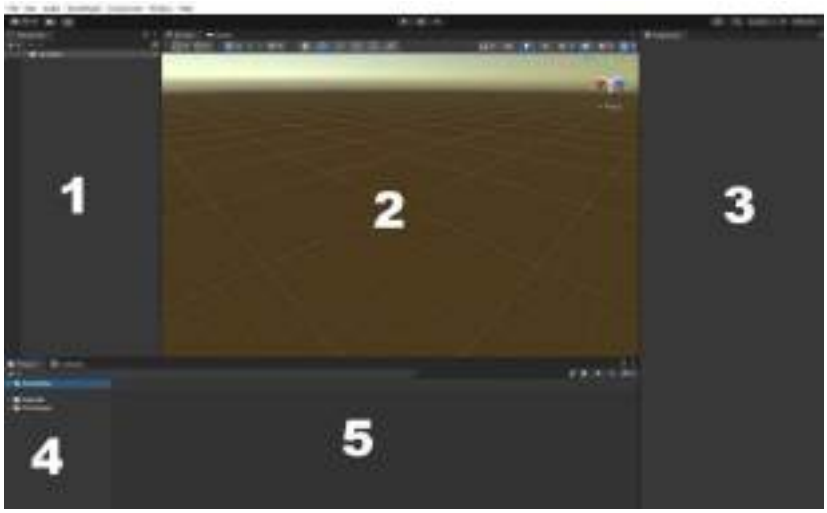
1. PRIMER ACCESO

Existen dos versiones de Unity, una versión gratuita y una versión profesional. Este manual se basará en las funcionalidades accesibles a los principiantes con la versión gratuita. La versión gratuita se puede encontrar en la página web de Unity. Una vez instalada la aplicación, podemos acceder a una demo donde podremos ver las capacidades de Unity y permitir a los nuevos usuarios descubrir las funcionalidades que ofrece estudiando las creaciones de sus desarrolladores.



2. INTERFAZ DE USUARIO

En esta parte del manual veremos de qué se compone la interfaz de usuario de Unity. Hay principalmente 5 áreas de la interfaz de Unity, listadas en la imagen de abajo:


1
2
3
4
5

- Vista jerárquica. Donde tendrán todos los objetos de la escena actual.
- Vista de escena Esta es el área donde construimos visualmente cada escena. La forma más sencilla de utilizar la vista de escena sería arrastrar objetos desde la vista de gestión de objetos a la vista de escena, lo que colocará el objeto en la escena; entonces podrás posicionarlo, escalarlo y rotarlo sin salir de la vista de escena. La vista de escena es también el lugar donde se editan los terrenos (esculpiéndolos, pintando texturas y colocando elementos), se colocan luces y cámaras y otros objetos.
- Vista Inspector. Esta vista tiene varias funciones dependiendo de lo que el usuario seleccione, si quiere las características de un objeto o la configuración del terreno. Si seleccionas "cámara" te mostrará las propiedades donde podrás personalizar características como la rotación, la posición o la escala.
- Vista del proyecto o gestión de objetos. Es la biblioteca de nuestro proyecto, similar a una biblioteca de herramientas y/o objetos. Puedes importar objetos 3D de diferentes aplicaciones a la librería, puedes importar texturas y crear otros objetos que puedes almacenar para usarlos en tu proyecto. Un proyecto normal contendrá varias escenas y una gran cantidad de activos, por lo que es necesario estructurar la biblioteca en diferentes carpetas que harán que los activos estén organizados y sean más fáciles de usar.
- Vista de búsqueda. Sirve para localizar objetos que necesitamos utilizar en la realización del proyecto.

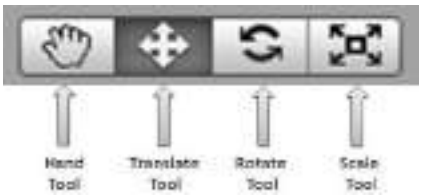
3. MENÚ DE APLICACIONES

A continuación se muestra el menú de opciones de Unity situado en la parte superior izquierda de la pantalla. A lo largo del manual se mostrarán las utilidades de cada una de las secciones de este menú.

3.1 BOTONES DE CONTROL

Debajo de las opciones de visualización verás una fila con 4 botones que puedes usar Q, W, E, R para alternar entre cada uno de los controles, que se detallan a

continuación:



- Herramienta Mano (Q): Este control nos permite movernos por la vista de la escena.

o ALT en permite nosotros a girar.

o CTRL permite nosotros a hacer zoom en y e n out

o SHIFT aumenta la velocidad de movimiento al utilizar la herramienta.

- **Herramienta Traslación (W):** Nos permite mover cualquier objeto seleccionado en la escena en los ejes X, Y y Z.

- **Herramienta Rotar (E):** Nos permite rotar cualquier objeto seleccionado en la escena.

- **Herramienta Escala (R):** Nos permite escalar cualquier objeto seleccionado en la escena.

Al utilizar las herramientas de traslación, rotación o escalado veremos un gizmo alrededor del objeto seleccionado con líneas para cada uno de los ejes. Podemos utilizar este gizmo para realizar operaciones de traslación, rotación y escalado.

3.2 BOTONES DE REPRODUCCIÓN

En Unity puedes ejecutar tu juego sin salir del editor, lo que es una gran ayuda para los diseñadores que construyen niveles y los desarrolladores que añaden nuevas mecánicas de juego.

La imagen muestra los controles de reproducción, situados en la parte superior del editor. Puedes entrar en la vista de juego en cualquier momento pulsando el botón de reproducción para ver el estado del proyecto (primero por la izquierda), hacer una pausa con el botón de pausa (centro) o avanzar con el botón derecho. Puedes jugar desde la vista de juego o ampliarla al máximo.

3.3 GUIONES

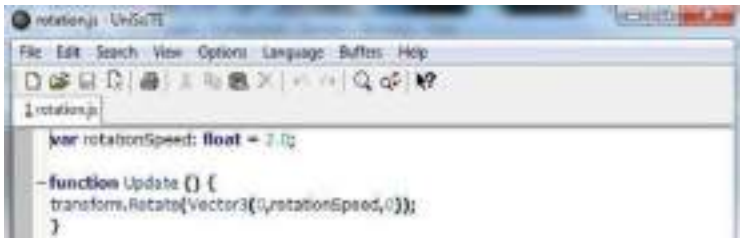
Los scripts permiten definir la lógica del juego. A lo largo del tutorial verás varios scripts sencillos como definir un movimiento, encender o apagar luces. El funcionamiento es

muy sencillo, primero se define el comportamiento deseado en un script y este se añade como componente al objeto al que se le quiere asignar ese comportamiento, incluso se puede añadir un script a un objeto vacío. Existen 3 funciones básicas para los scripts:

- Función Update( ): función que define el bucle principal del juego de forma que se ejecuta una vez por cada fotograma que se renderiza.

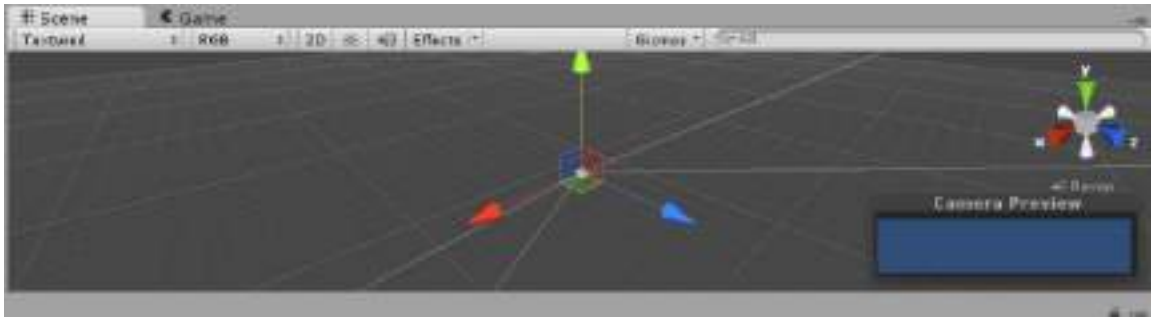
- Función Awake( ): Las variables definidas en esta función se inicializarán justo antes de que comience el juego. Esta función se ejecutará una sola vez mientras dure la instancia de script.
- Función Start( ): Esta función es muy similar a "Awake( )" pero tiene una diferencia vital; sólo se ejecutará si la instancia del script está habilitada. Esto nos permitirá retrasar la inicialización de algunas variables de estado del juego. También debemos tener en cuenta que la función "Start()" siempre se ejecutará después de que la función "Awake()" haya finalizado.

Unity 3D también tiene un editor de scripts por defecto llamado uniscite.



4. INICIO DE UNITY 3D

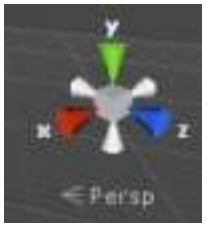
Lo primero que hay que hacer es iniciar Unity 3D. Cuando hagas esto se cargará un proyecto en blanco. Si quieres otra escena entonces buscamos en el menú "File -> Open Scene" y buscamos en el directorio del proyecto ("Assets/scenes"). Modos de visualización por defecto la vista de escena tiene una perspectiva 3D de la escena.



Perspectiva

Puedes cambiar la vista de la escena a varias vistas: de arriba abajo, lateral y frontal.

En la parte derecha de la vista de la escena verás un "Gizmo" que parece una caja de la que salen conos.



- Con este Gizmo podemos cambiar la perspectiva de la vista.
- Al hacer clic en el recuadro, pasará al modo perspectiva.
- Haciendo clic en el cono verde (y) pasará al modo Top-Down.
- Si hace clic en el cono rojo (x), pasará al modo Lateral (derecha).
- Haciendo clic en el icono azul (z) accederá al modo Frontal.
- También tenemos 3 conos grises que nos llevarán a las siguientes modalidades:
- "Atrás, Izquierda y Abajo".

4.1 VISTA JERÁRQUICA

La vista de jerarquía contiene todos los elementos que componen la escena del juego. Puedes manipular todos los objetos de la escena desde esta vista, cada vez que introduces un elemento en la escena, se añade una entrada para este elemento en esta vista. Seleccionar un objeto en esta vista también lo selecciona en la escena y en el inspector. Esto permite y facilita mover, escalar, rotar y eliminar el objeto o editar sus parámetros. Todo ello sin tocar una sola línea de código.

Para insertar un nuevo objeto en la escena puedes hacer clic en "CREAR" y se mostrarán todos los objetos que quieras añadir.


Hierarchy
Create
First Person Controller
Main Camera
Point light
Point light
Point light
Point light
Terrain
Particle System
Camera
GUI Text
GUI Texture
3D Text
Directional light
Point light
Spotlight
Area Light
Cube
Sphere
Capsule
Cylinder
Plane
Quad
Sprite
Cloth
Audio Reverb Zone
Terrain
Ragdoll...
Tree
Wind Zone

4.2 VISTA DE LA ESCENA

La vista de escena es la parte del entorno gráfico 3D para crear cada escena. Para trabajar en la vista de escena lo más sencillo es arrastrar un objeto desde la vista de proyecto a la vista de escena, lo que colocaría el objeto en la escena y, por tanto, lo situaría donde usted desee.

La vista de escena es también el lugar donde editar terrenos (esculpirlos, pintar texturas y colocar elementos), colocar luces y cámaras y otros objetos. A continuación se explican las opciones de la vista de escena:

En la parte izquierda de la vista de escena encontrarás un conjunto de botones para cambiar la configuración general de la pantalla. Veamos cada uno de ellos, de izquierda a derecha:

Modo de renderizado

El valor por defecto será "Texturizado". Si se pulsa, aparecerá una lista desplegable con distintas opciones de renderizado:

- Texturizado: Las texturas se renderizan en la vista.
- Malla metálica: Las superficies no se renderizan, sólo vemos la malla.
- Alambre texturizado: Las texturas se renderizan, pero también vemos la malla.
- Render Paths: cambia las representaciones y características de los objetos en función de los proyectos que se estén creando.
- Resolución lightmap: para dar una resolución más óptima a los objetos que componen la escena.

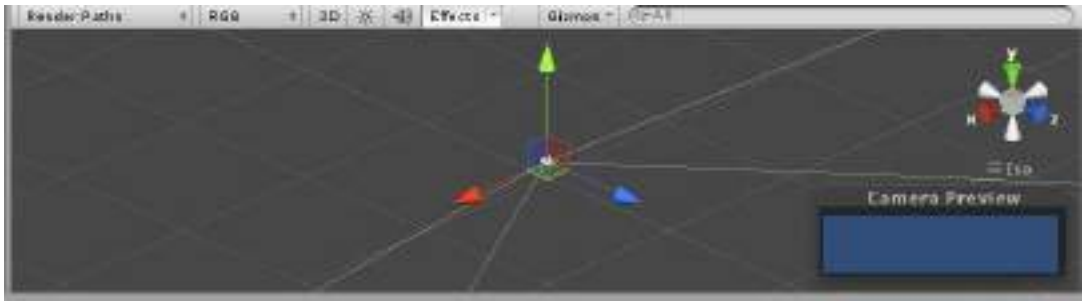
Modo de color, mostrado como "RGB".

Si haces clic en este botón, aparecerá una lista desplegable con los modos de color disponibles y podrás ver las distintas formas de visualizar la escena:

- RGB: Se representan todos los colores.
- Alfa: El modo cambia a "Alfa".
- Sobretrazado: El modo cambia a "Sobretrazado".
- Mipmaps: El modo se cambia a "Mipmaps".

Opción 3D-2D

Donde cambiamos la visualización de la escena de un plano X, Y, Z a un plano de visualización X, Y.



Interruptor de la luz

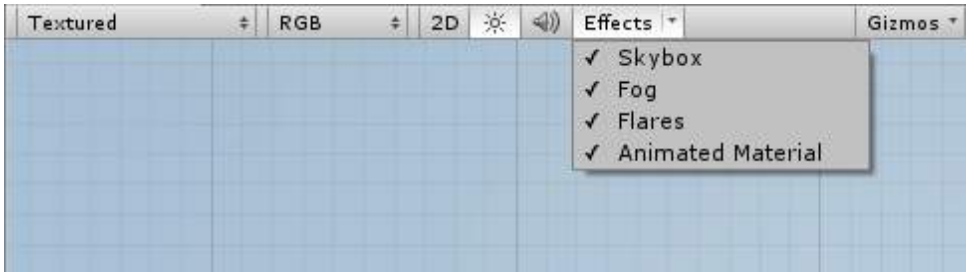
El siguiente botón enciende o apaga la iluminación del escenario.

Audio

El siguiente botón activa o desactiva el audio de la escena.

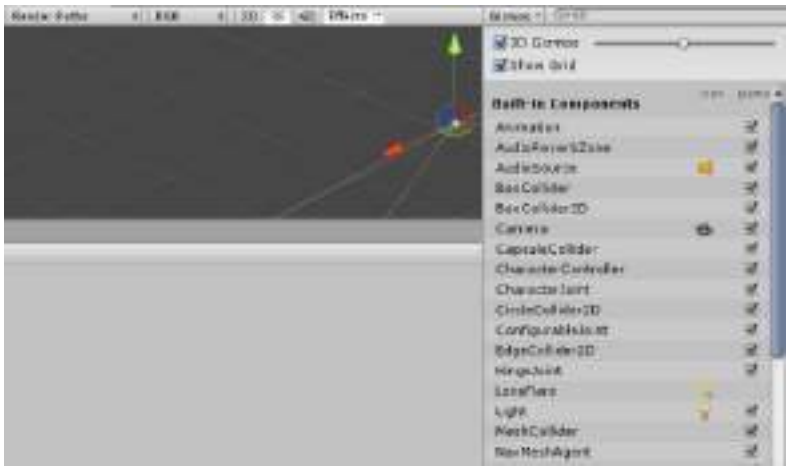
Efectos

El botón de efectos se utiliza para desactivar los efectos de vídeo con el fin de mejorar el rendimiento durante el diseño del proyecto.



Artilugios

Ayudan a perfeccionar los gráficos de la escena incluso cuando se utilizan lenguajes de programación.



4.3 VISTA DEL JUEGO

La vista de juego te permite previsualizar el juego en la vista tal cual o maximizándolo a pantalla completa. En esta vista puedes ver el juego en movimiento para comprobar que funciona correctamente.

La vista Juego es muy útil pero tiene un inconveniente, al probar el proyecto en el ordenador, se mueve con las teclas o el ratón lo que no ayuda mucho si el proyecto se está desarrollando para otra plataforma, por ejemplo un dispositivo móvil. En este caso

las pruebas se tienen que hacer usando un emulador o compilando una versión móvil.

- El botón "Maximizar al jugar": permite maximizar la vista del juego a pantalla completa.
- El botón "Estadísticas":
- El botón "Gizmos":
- Desplegable "Aspecto libre":

4.4 OPINIÓN DEL INSPECTOR

En la vista del inspector se muestran los parámetros de los objetos seleccionados en la vista de escena o en la vista jerárquica. De este modo, cuando seleccionas un objeto en la escena, puedes modificar sus parámetros en el inspector (posición, rotación, traslación, etc.). El inspector también sirve como panel de herramientas para algunos elementos como los terrenos, permitiéndote esculpirlos, añadir texturas y mucho más.

En la vista del inspector se pueden ver los elementos que definen el comportamiento de los objetos, estos elementos se denominan componentes. Además, aquí se pueden activar o desactivar los objetos y sus componentes asociados. Los componentes pueden ser scripts, animaciones, colliders...etc.

Por último, el inspector cambia en función del elemento seleccionado.

4.5 VISTA DEL PROYECTO

La vista del proyecto es una biblioteca de activos para la creación del juego. En esta vista se almacenan todos los elementos que se crean e importan para ser utilizados en el juego. Estos elementos pueden ser objetos 3D, texturas, sonidos o scripts.

En esta ventana es posible disponer de multitud de recursos, pero esto no implica la utilización de todos ellos para la creación del videojuego.

Dado que esta vista contiene tanto los elementos utilizados en la escena como los que no, puede contener una gran cantidad de datos. Esto implica que hay que mantener una buena estructura para facilitar el trabajo del usuario. La organización de esta vista puede hacerse creando una jerarquía de carpetas. El desplegable "crear" contiene las opciones que permiten una buena organización de la vista.

5. CREAR UN NUEVO PROYECTO

Crear un nuevo proyecto implica ir a la barra superior de Unity y hacer clic en Archivo

-> Nuevo Proyecto para abrir la ventana de diálogo "Crear nuevo Proyecto".

Seleccione primero la carpeta en la que desea guardar el proyecto haciendo clic en "Examinar...". Para crear un nuevo proyecto, tienes que importar los paquetes necesarios para el proyecto.

Los paquetes son conjuntos de activos y pueden importarse al crear el proyecto o en cualquier momento posterior.

Los paquetes importados pueden ser los predefinidos en Unity, otros creados por el usuario u obtenidos de la Web, algunos de descarga gratuita y otros de pago.

En la lista que aparece en el sitio web, seleccione los activos y haga clic en "descargar".

Para los activos que se encuentran en el programa, seleccionamos los que queremos y hacemos clic en "Crear". El paquete básico a importar es el conjunto estándar de activos.

Cuando se crea el nuevo proyecto, Unity 3D se reinicia creando la estructura del proyecto en la carpeta especificada. Se abre la pantalla de inicio de Unity 3D, mostrando los activos importados en la vista de proyecto y una cámara en la vista de escena y jerarquía.

Algunos de los paquetes más importantes son:

- Controlador de personaje: Permite crear scripts para mover al personaje.
- Partículas: Permite crear partículas como fuego, humo, cascadas y explosiones.
- Skyboxes: Permite la creación de cielos.
- Terreno: Permite crear y esculpir terrenos.
- Agua: Crea agua (mar, ríos, etc).
- Árbol: Creador para crear árboles.

5.1 CREAR UNA NUEVA ESCENA

Unity 3D funciona por escenas y cuando creamos un nuevo proyecto automáticamente se crea una nueva escena, si queremos crear otra escena, vamos a File _ New Scene y la guardamos automáticamente en File _ Save Scene. La escena se guarda en la carpeta asset del proyecto y aparecerá en la vista del proyecto (las carpetas se encuentran en Documents/New Unity Project o como queramos llamar a la escena).

Una vez guardada la escena podemos crear todos los elementos que la compondrán,

y disponemos de diferentes formas de manipular la escena.

Algunos trucos para la manipulación de escenas:

- Si queremos ver la escena desde distintos ángulos podemos rotarla usando ALT y moviendo el ratón.

- Si quieres acercar o alejar la imagen, puedes utilizar la rueda del ratón.
- Las flechas del teclado permiten mover la escena a izquierda, derecha, arriba y abajo. Lo mismo puede hacerse manteniendo pulsado el ratón y moviéndolo en una dirección u otra.
- Para aumentar la velocidad de movimiento de la escena mantén pulsado SHIFT.

Al crear una escena, aparece por defecto una cámara en nuestro mundo. Si hacemos clic en "Cámara principal", las características de la cámara se muestran en el inspector y se pueden modificar (posiciones, rotación, escala).

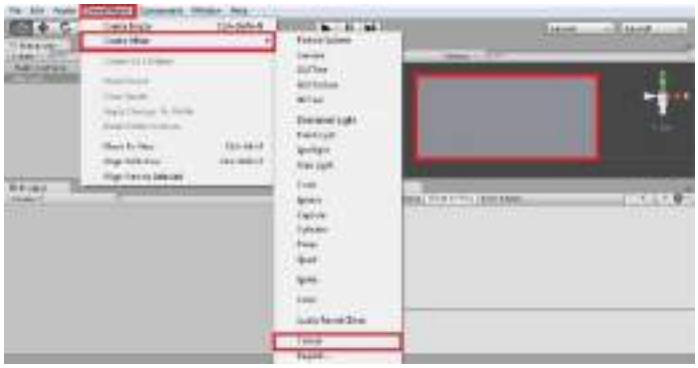
La cámara define el punto de vista del jugador y se pueden combinar varias cámaras en la misma escena.

5.2 CREAR EL TERRENO

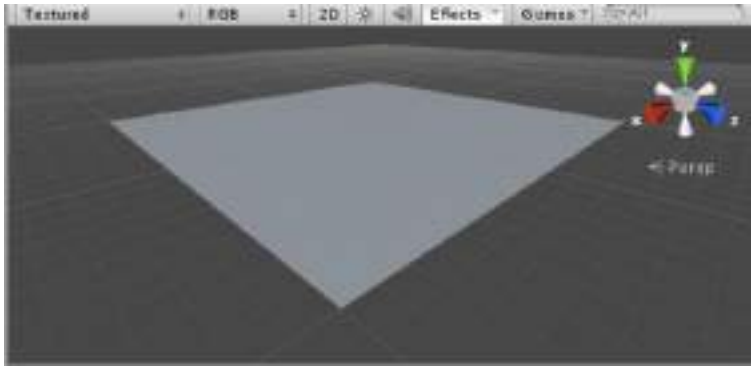
Unity 3D tiene una herramienta con la que puedes definir cualquier terreno, normalmente maneja los terrenos como una malla plana, pero puedes texturizar y esculpir sin salir del editor. Desde el menú puedes crear un terreno haciendo clic en:

- "Terreno -> Crear terreno".

Obtienes un terreno plano, si el terreno no es visible tienes que desactivar las luces en la vista de escena.



Una vez creado el terreno, se obtiene el siguiente resultado:

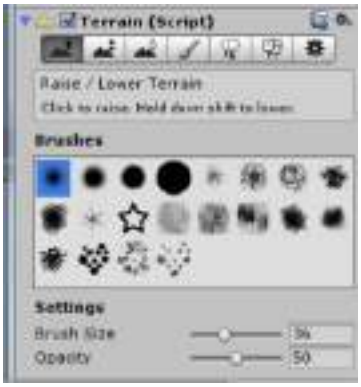


Si queremos personalizar el terreno y hacerlo más realista tenemos que hacer uso de la herramienta de terreno. El terreno se selecciona en la vista de escena o en la vista de jerarquía para que aparezca la herramienta de edición del terreno en la vista de inspector.

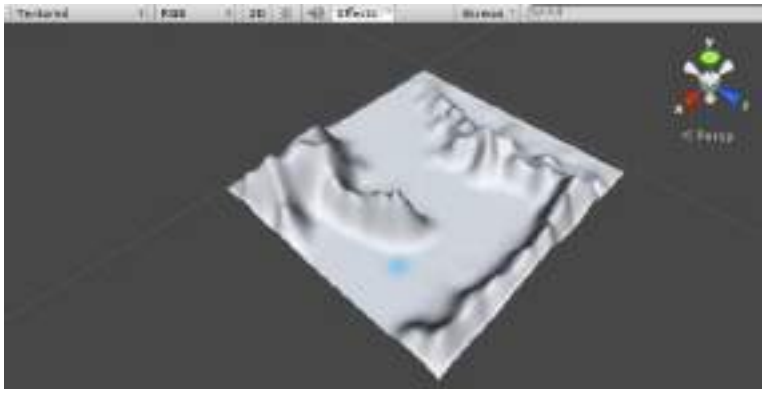
Esta herramienta se divide en 3 partes:

- Transformar: Permite mover, rotar y escalar el terreno en los ejes x,y,z.
- Script de terreno: Contiene varias herramientas y propiedades para el terreno que se explicarán más adelante.
- Colisionador del terreno: Contiene las propiedades de colisión del terreno.

- Vamos a proceder a detallar el panel "Script de terreno" y veremos una fila de botones. Estos botones del editor de terreno te permiten realizar diferentes tareas. La descripción de lo que los botones le permiten hacer de izquierda a derecha:
- Subir / Bajar: permite subir y bajar la geometría del terreno mediante un pincel.
- Fijar altura: pintamos el terreno con un límite de altura.
- Alisar: permite alisar un terreno para eliminar las esquinas.
- Pintar textura: permite pintar texturas en la superficie del terreno.
- Colocar árboles: permite colocar árboles.
- Pintar detalles: permite dibujar detalles del terreno, como la hierba.
- Configuración del terreno: Acceso a las propiedades del terreno donde podemos cambiar varias propiedades.

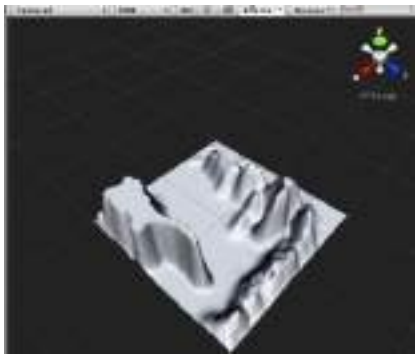


Primero define las alturas del terreno haciendo clic en los lugares donde quieras colocar una montaña, cuanto más tiempo mantengas pulsado el ratón más grande se hará la montaña. El terreno puede transformarse en la siguiente imagen:



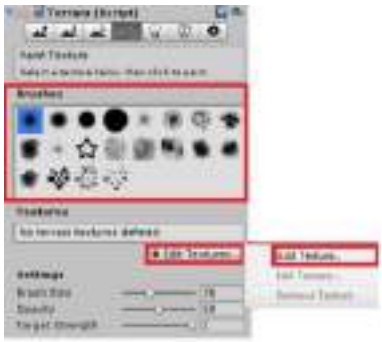
La segunda opción "Fijar altura" permite establecer una altura máxima en el campo "Altura".

Fijando una altura máxima, el terreno puede modelarse aún más para darle la geometría adecuada. Una vez alcanzada la altura máxima, el terreno se aplana. La tercera opción "Suavizar" permite suavizar los picos del terreno. Es aconsejable esculpir por etapas, primero las partes grandes y luego afinar los detalles más pequeños. Al final se obtiene un terreno como el que se muestra en la imagen:



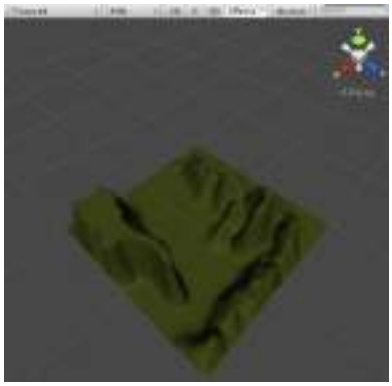
Una vez obtenido un terreno con la apariencia deseada, se pueden aplicar varias texturas. La primera textura se considera la textura base y se aplica a todo el terreno. Las demás texturas se aplicarán en capas superiores.

La cuarta opción "Pintar la textura del terreno" le permite añadir textura al terreno. Elija primero el tipo de pincel que desea utilizar en "Pinceles" y, a continuación, añada la textura elegida en "Texturas" haciendo clic en "Editar textura> Añadir textura".



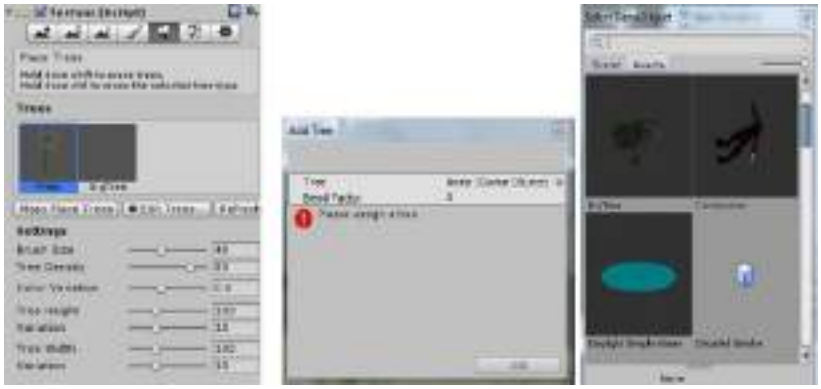
Aparece una ventana "Añadir textura del terreno", haz clic en el círculo rojo resaltado para obtener la ventana "Seleccionar textura 2D", puedes elegir una textura y aparecerá en el campo "Splat" y en el campo "Textura" del inspector. También puede elegir la anchura y la longitud de la textura.

Una vez aplicadas las texturas, aparecerán en el terreno, que tendrá el aspecto que se muestra en la siguiente imagen:



Unity 3D ofrece un soporte especial para introducir árboles. Puedes añadir tantos árboles como quieras utilizando la quinta opción "Colocar árbol". Primero utilizas la opción "Editar árboles" para elegir el árbol que quieres añadir a tu mundo, después personalizas el árbol definiendo varios parámetros, el tamaño del pincel con el que se va a colocar el árbol, la densidad de los árboles (determina el número de árboles que se van a colocar), la variación de color (diferencia de color entre un árbol y otro), la altura y la anchura del árbol.

Por último, haz clic en cualquier parte del terreno para añadir el árbol en el lugar elegido. Para eliminar los árboles, mantenga pulsada la tecla Mayús y haga clic en el terreno.

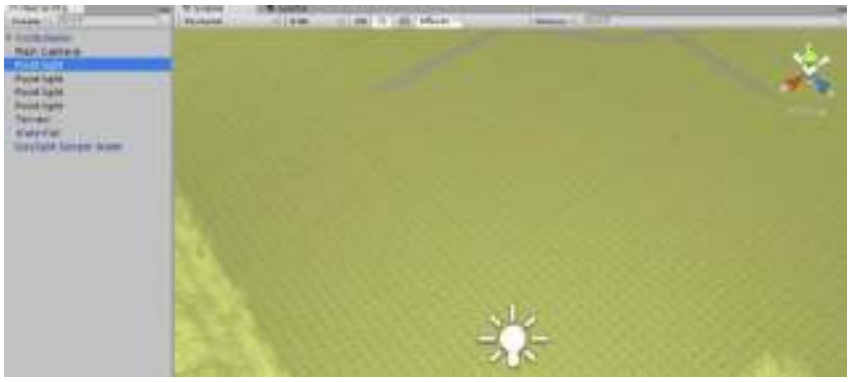


También puedes añadir varios árboles a tu mundo en bloque para crear un bosque de una sola vez. Haz clic en "Terreno->Colocar árboles en masa" y aparecerá un cuadro de diálogo en el que podrás determinar el número de árboles que quieres introducir en el juego.

Se puede seleccionar un árbol yendo a la carpeta "Activos estándar > Creador de árboles > Texturas", hacer clic y arrastrarlo al terreno.

5.3 ILUMINACIÓN

Si quieres iluminar el escenario con la misma intensidad sólo tienes que añadir luz puntual y poner las luces deseadas en el escenario: créate>luz puntual



Si se desea una iluminación de alta calidad, Unity integra la tecnología Beast, que es un software de Autodesk que genera mapeados de iluminación de alta calidad.

"Crear mapa de luz" se utiliza para diferentes creaciones de luz que permiten variar la iluminación de las diferentes texturas del terreno.

Acceda a Ventanas>Mapa de luz y vea lo siguiente:

Para controlar la resolución de los mapas de luz, el valor de resolución se establece en el panel Bake y se aplica automáticamente.

Lightmapping mejora la definición sobre el terreno como zona de árboles y sus sombras.

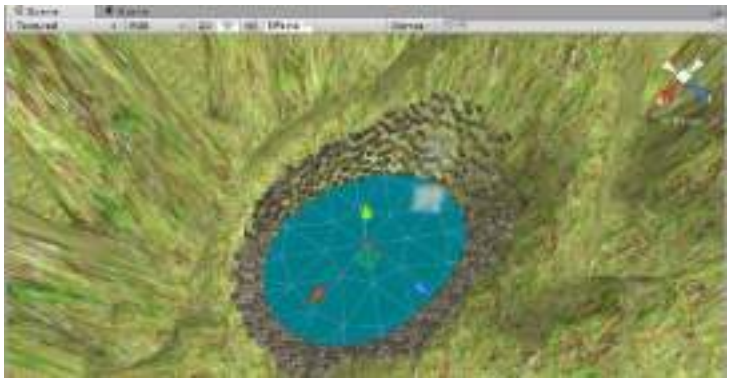
5.4 MEDIO TERRESTRE

El entorno del terreno ayudará al usuario a personalizar el entorno en el que se encuentra la escena que está construyendo, por ejemplo, añadiendo un cielo o agua.

5.4.1. AÑADIR AGUA A LA ESCENA

Unity incluye varias formas de añadir agua a la escena, una de las cuales es la que el usuario puede utilizar directamente. Dos de estas directamente son el "Bake de lightmapping" y el "Nighttime Simple Water".

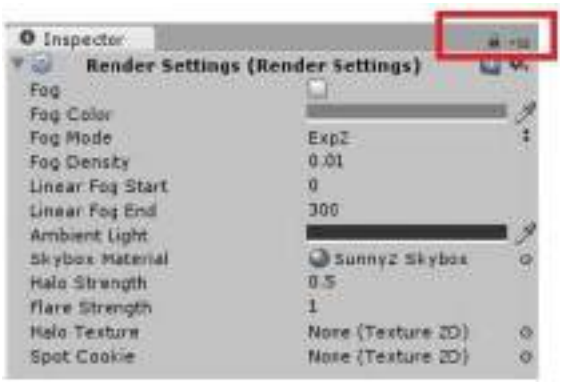
Como ejemplo añadiremos "Daylight Simple Water" dentro del terreno. Vaya a "Standard asset > Water", seleccione el "Daylight Simple Water" y arrástrelo a la escena.



Las características del agua pueden verse en el inspector:

5.4.2 AÑADIR SKY

Puedes añadir el cielo de dos formas, adjuntando un skybox a la cámara o directamente a la escena. Vaya a "Edición > Ajustes de renderizado" para activar los campos del inspector como se indica a continuación:



Este menú le permite configurar el cielo de la escena, los efectos de niebla y algunos efectos de iluminación global. Puedes importar tus propias texturas o utilizar las que vienen por defecto en los assets de unity.

Hay 2 opciones para aplicar el cielo a la escena:

Opción 1: ve al menú Render Settings, haz click en el pequeño icono de disco a la

derecha del campo Skybox Material, se abrirá una nueva ventana con todos los recursos materiales cargados en el proyecto.

Opción 2: en la vista del proyecto, en "Standard Asset > Skyboxes", seleccione el cielo apropiado para su escena y arrástrelo al "Sybox Material". Para este resultado se ha elegido el cielo "Sunny 1 Skybox".

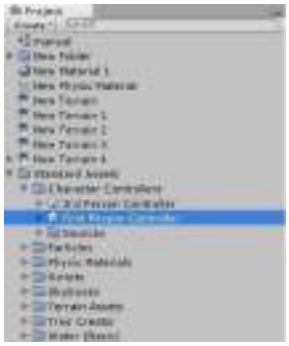
Unity 3D ofrece varias opciones para personalizar el entorno de la escena, estas son algunas de ellas:

- Niebla: permite añadir niebla a la escena, la escena se vuelve brumosa.
- Color de la niebla: el color de la niebla, por defecto es gris.
- Modo de niebla: lineal, exponencial o exponencial al cuadrado. Esto controla cómo se desvanece la niebla.
- Densidad de la niebla: densidad de la niebla, sólo se utiliza para exponencial y exponencial al cuadrado. a. Una mayor densidad disminuirá la visibilidad y una menor densidad aumentará la visibilidad.
- Inicio/final de niebla lineal: inicio y final de las distancias de niebla, sólo se utiliza si está activada la opción de niebla lineal.
- Luz ambiente: por defecto es un gris. Si queremos dar a la escena un esquema de iluminación diferente, cambiar la Luz ambiente es un buen comienzo.
- Skybox Material: la textura del cielo de la escena.
- Fuerza del halo: Tamaño de todos los halos de luz en relación a su alcance.
- Intensidad de la bengala: Intensidad de todas las bengalas de la escena.
- Textura Halo: referencia a una textura que aparecerá como el brillo de todos los halos en las luces.
- Cookie de foco: La referencia a una Texture2D que aparecerá como máscara de cookie para todos los focos.

5.4.3 MOVIMIENTO

Unity incluye un controlador en primera persona que facilita la creación de juegos con vista en primera persona. Accediendo a "Standard Assets > Character Controllers", seleccione la opción

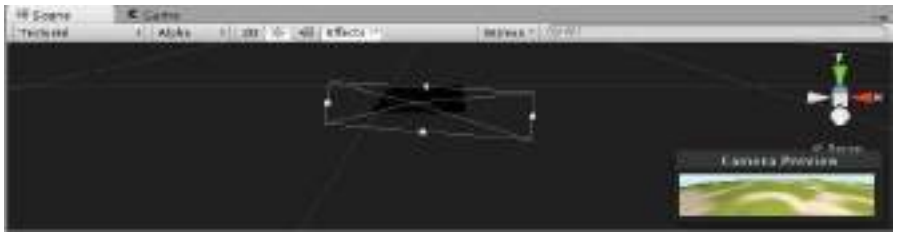
"Controlador en primera persona" y arrástralo hasta la posición donde quieras colocarlo en la escena.



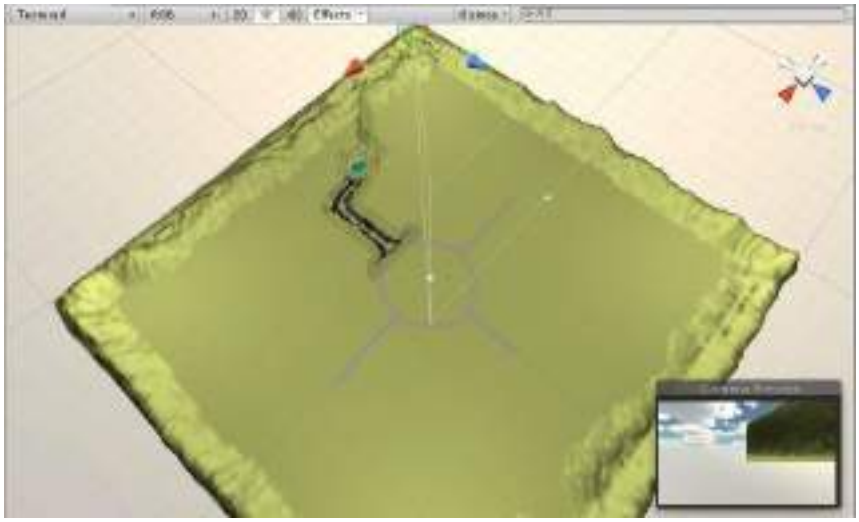
En versiones anteriores de Unity sólo se podía programar el movimiento con un pequeño script que se muestra a continuación:

```
var velocidad=8;
public var ptosalida:GameObject;

function Update () {

        var direccion=transform.TransformDirection( Vector3( 0, 0, 0.3 ));

        if( Input.GetKey("up")){
                //transform.Translate( 0 , 0 , 0.1 );
                rigidbody.AddForce( direccion*velocidad );
        }

        if( Input.Getkey("left"))
                transform.Rotate( 0 , -1 , 0 );

        if( Input.Getkey("down")){
                //transform.Translate( 0 , 0 , -0.1 );
                rigidbody.AddForce( direccion*velocidad*-1 );
        }

        if( Input.Getkey("right"))
                transform.Rotate( 0 , 1 , 0 );
}
```

Puedes ver la cámara que se ha añadido a tu escena, desde los cuatro puntos de cámara puedes aumentar o disminuir el enfoque que tendrá la cámara.

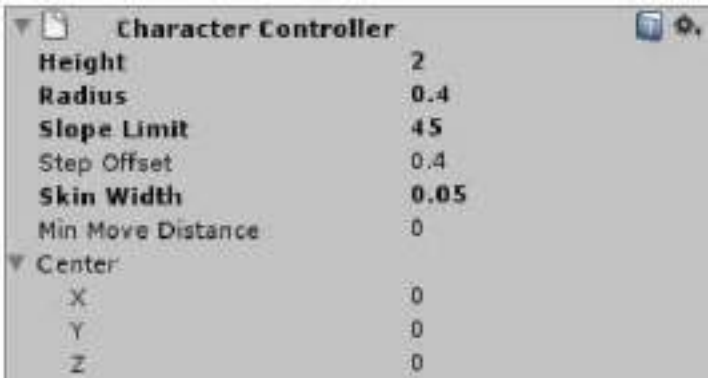


Una vez que tenemos nuestra cámara lista, el movimiento se puede hacer a través de varias opciones: Movimiento: con flechas o teclas (W: adelante, A: izquierda, D: derecha y S: atrás). La cámara se controla con el ratón.



El salto con la tecla espacio.

El "Controlador en primera persona" puede personalizarse del mismo modo que los demás elementos del inspector. Para ello, se selecciona en la vista de jerarquía y sus características aparecen en el inspector.



Explicación de los controladores:

- Altura: altura del controlador
- Radio: radio controlador
- Límite de pendiente: valor máximo de la pendiente que puede subir el regulador.
- Desplazamiento del paso: este valor se sitúa entre 0,1 y 0,4 para un ser humano de 2 metros de tamaño.
- Anchura de la piel: permite controlar los colisionadores del personaje para que

no se bloquee.

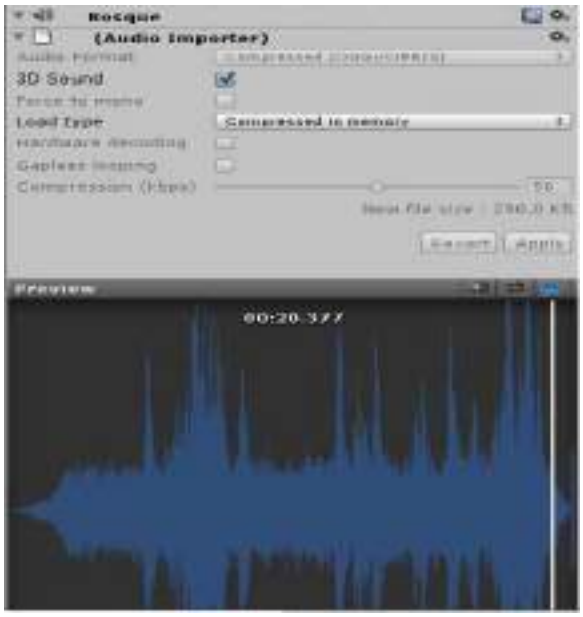
- Distancia mínima de movimiento: Si el personaje intenta moverse por debajo del valor indicado, no se mueve en absoluto. Esto se puede utilizar para reducir las vibraciones. En la mayoría de las situaciones este valor debe dejarse en 0.
- Centro: la posición del controlador, sus coordenadas x, y, z.

5.4.4 AUDIO EN LA ESCENA

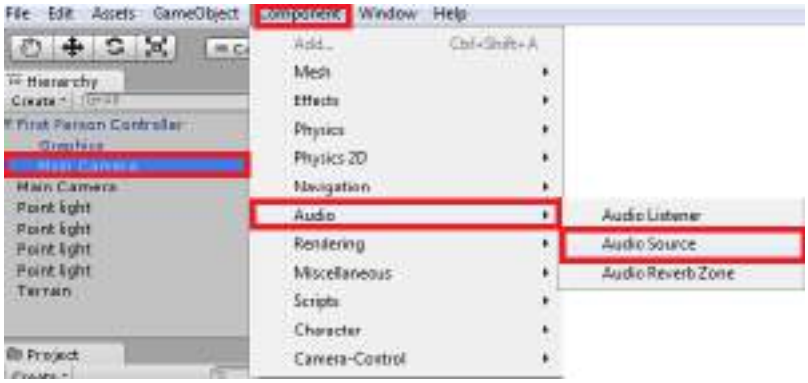
En podemos insertar sonido en el juego. Hay dos tipos diferentes de audio, audio 3D o audio 2D.

El audio 3D se escucha en una posición y se atenúa a medida que te alejas de esa posición. El audio 2D se escucha uniformemente en toda la escena.

Primero crea una carpeta llamada "sonido" en la que guardes un sonido importado, por ejemplo uno que represente los sonidos ambientales de una selva o un bosque.



Seleccione la "Cámara principal" y vaya a Componente> Audio> Fuente de audio.



A continuación, el sonido importado se arrastra a la casilla "Clip de audio" de la vista del inspector y se configura el audio:

La casilla "Reproducir al despertar" se activa para que el sonido comience al principio del proyecto.

La casilla "Bucle" se activa para que la música se reproduzca en bucle y se repita tantas veces como sea necesario.

En el gráfico "Listner" del componente, seleccione el punto verde del punto más alto y arrástrelo hasta el centro de coordenadas (0,0).

En "Distancia mínima" se define una distancia mínima de 0,1.

5.4.5 ENCENDER Y APAGAR LA LUZ DEL ESCENARIO

En esta versión de Unity encender y apagar la luz del escenario funciona haciendo clic en el botón de luz que se encuentra en las opciones de vista del escenario, pero si quieres apagar la luz en el escenario que crees en algún momento, puedes escribir un simple script como se muestra a continuación:

```
var myLight : Light;
var myLight1 : Light;

function Start(){
    myLight = Light.FindObjectOfType(typeof(Light));
    }


function Update () {

    if(Input.GetButtonDown('Fire1')){
        myLight.enabled = !myLight.enabled;
        myLight1.enabled = !myLight.enabled;
    }
}
```

Explicación del guión:

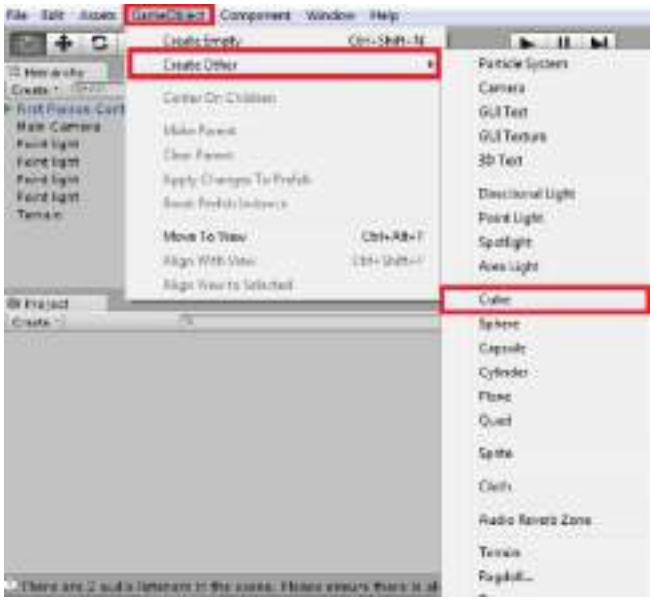
- myLight = Light.FindObjectOfType(typeof(Light)); _ busca en la escena el objeto de tipo "luz" (la luz de la escena).
- if(Input.GetButtonDown("Fire1")) cuando se pulsa el botón izquierdo del ratón.
- myLight.enabled = !myLight.enabled; _ cambia el estado de la luz, si está encendida, se apaga y viceversa.
- myLight1.enabled = !myLight.enabled _ esta línea está duplicada porque hay 2 luces en nuestra escena.

Una vez creado el script, se añade al objeto y en los parámetros que aparecen en la vista del inspector, se arrastra cada una de las 2 luces para que quede así:

5.4.6. CREACIÓN DE UN OBJETO

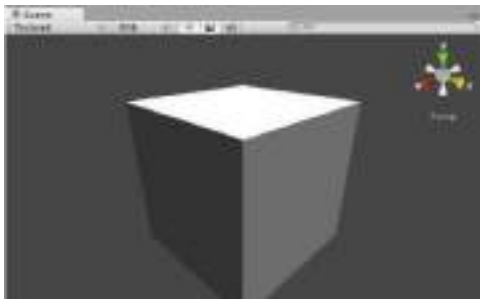
Explicaremos cómo crear un elemento, en este caso un cubo.

Para crear los objetos predefinidos en Unity 3D, vaya a GameObjet > Create Other y aparecerá una lista de objetos.



Algunos componentes importantes de GameObject son:

- Cámara: permite insertar una cámara en la escena, aunque al crear un proyecto Unity coloca una cámara por defecto.
- Luz direccional: introduce una luz direccional uniforme que permite ver la escena.
- Cubo: coloca un cubo en la escena.
- Esfera: introduce una esfera
- Cápsula: introducir una cápsula
- Cilindro: poner un cilindro
- Plano: introduce un plano en la escena. Cuando se selecciona uno de los "GameObjects", aparece en la vista de escena, en la vista de juego y en la vista de jerarquía. Además, en la vista del inspector se pueden ver los atributos del objeto para la parametrización del mismo.

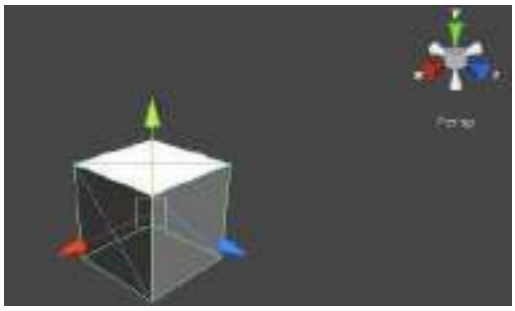


5.4.7 MANIPULACIÓN DEL OBJETO

Una vez creado el objeto, en este caso un cubo, podemos modificarlo para que tenga las dimensiones, posición y forma necesarias para conseguir las características deseadas.

La posición del cubo puede modificarse con el segundo botón de control:

Como podemos ver en la siguiente imagen, podemos mover el cubo en los ejes X (rojo), Y (verde) y Z (azul). Para mover el cubo basta con pulsar sobre el eje donde queremos moverlo y arrastrarlo hasta la nueva posición.



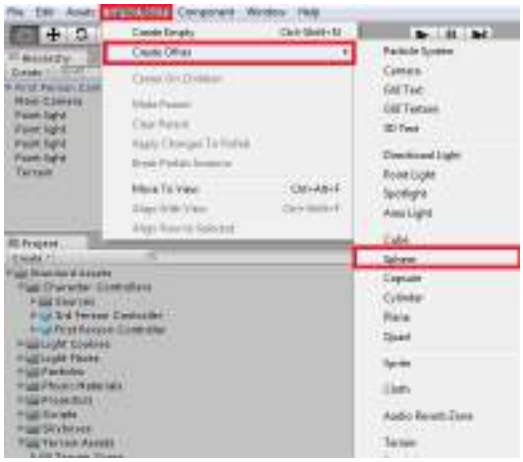
El tercer botón de control te permite girar el cubo.

Comparando la imagen anterior, en ésta vemos cómo gira el cubo.

Los distintos círculos que aparecen alrededor del cubo cuando se selecciona la opción de rotación permiten al usuario elegir el eje sobre el que desea realizar esta operación. Para girar un elemento, haga clic sobre el identificado con círculos de colores y mueva el ratón en la dirección seleccionada.

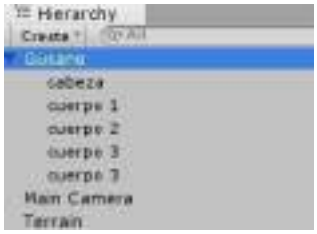
5.4.7 CREACIÓN DE UN GUSANO

Como ya sabemos crear un objeto, vamos a crear un personaje sencillo, un gusano basado en esferas. La primera esfera se crea en "GameObject > Create Other > Esfera":

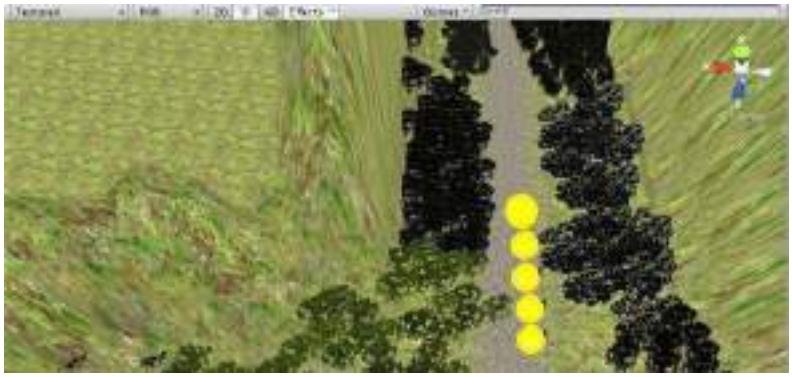


Coloca la esfera sobre el terreno en la posición que desees. Para crear otras esferas idénticas, pulsa "CTRL + D", esta nueva esfera aparece encima de la anterior y se puede mover. Repite la misma acción hasta que tengas 4 esferas. Puedes ver que las esferas aparecen en la vista jerárquica donde puedes renombrar la primera esfera a "cabeza" y las otras a "cuerpo". Para renombrar un Game Object, selecciónalo en la vista jerárquica, haz click derecho y "Renombrar". Modifica las esferas que representan el cuerpo haciéndolas más pequeñas utilizando el inspector para mayor precisión.

Además, para optimizar la organización de nuestro proyecto, se crea un objeto vacío en "GameObject > Create Empty", se renombra como gusano y se arrastran la cabeza y el cuerpo sobre el gusano.



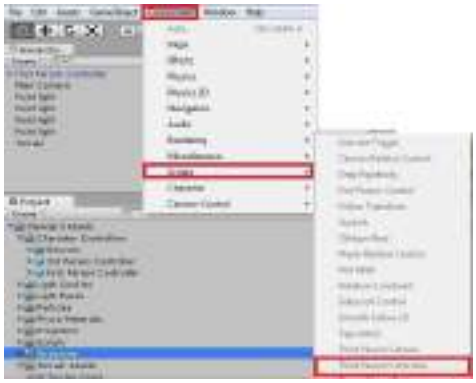
A continuación se importa una textura, se aplica al personaje y se obtiene el siguiente resultado:



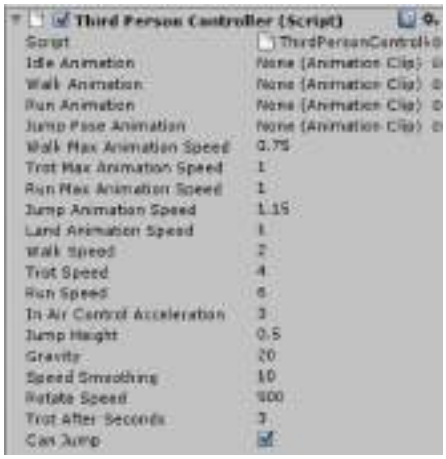
Por último, para hacer que todos los elementos que componen el cuerpo del gusano sigan a la cabeza, se utilizará el componente "Smooth follow", que se explicará en el siguiente apartado.

5.4.8 MOVILIDAD DEL OBJETO

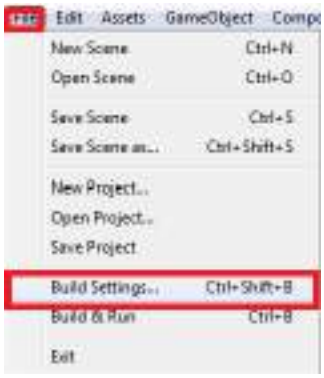
Selecciona la cabeza en la vista jerárquica, ve a Componente> Scripts> Controlador en tercera persona. De esta forma puedes dar movilidad a la cabeza y controlarla en tercera persona. Si quieres mover toda la serpiente, tendrías que aplicar el "Third Person Controller" a todo el gusano.



Y podemos ver en la vista del inspector como aparece el nuevo componente con todos los parámetros configurables.



Para finalizar el proyecto, podemos exportarlo y ejecutarlo:



Usted selecciona la plataforma y la arquitectura en la que desea exportar el proyecto, puede exportar en las plataformas WINDOWS, MAC, LINUX, ANDROID, IOS puede con la concesión de licencias libres si desea exportar a las consolas de juegos como XBOX, WII y PLAY STATION tendría que pagar por la licencia.

Por último, haz clic en Buid y Run y el proyecto empezará a exportarse, dale un nombre y guárdalo en la carpeta que elijas, terminando con un ejecutable de la plataforma que hayas elegido,

teniendo en cuenta la arquitectura del ordenador en el que se realiza el proyecto.

Al final del proyecto, se puede concluir que los motores de videojuegos son herramientas muy útiles que facilitan el trabajo de los creadores al permitirles centrarse en los aspectos creativos y no tanto en los estructurales, con el fin de satisfacer las demandas de los consumidores en lugar de adaptarse a consumidores exigentes.

Unity 3D es uno de los motores más actuales para el diseño de videojuegos, es muy completo y un motor muy potente. Además, en cada versión se ha ido adaptando a las nuevas tendencias como el diseño multiplataforma. Esta característica le da a Unity3D una ventaja sobre algunos motores de videojuegos.

Para llevar a cabo este proyecto hemos utilizado la versión gratuita de Unity 3D, pero no por ser gratuita es menos potente. Unity ofrece a los usuarios un entorno sencillo que permite crear un proyecto muy rápidamente y como no es necesario tener profundos conocimientos de programación cualquiera puede utilizarlo con unas nociones básicas.

Hay que tener en cuenta que Unity no sólo permite crear videojuegos o visitas virtuales, sino que también es una herramienta muy popular entre arquitectos y diseñadores, ya que permite crear entornos 3D de forma sencilla y rápida.

Unidad 3 - How to create codes using the C# programming language

INTRODUCTION:

En 2002 Microsoft anunció la aparición del lenguaje C#, similar a C++ y Java, pero mejorado (en

música el símbolo # significa "un semitono más alto"). Este desarrollo era parte importante de la iniciativa "punto net" de Microsoft, que describiremos a continuación. Debido a la similitud entre C# y sus predecesores, al aprenderlo los lenguajes C, C++ y Java le serán más comprensibles si alguna vez necesita utilizarlos.

- *C# es un lenguaje orientado a objetos, derivado de C++ y Java.*
- *El marco de trabajo (o framework) .NET de Microsoft es un producto importante, que incluye los lenguajes C#, C++ y Visual Basic.*
- *Los programas son listas de instrucciones que las computadoras obedecen de manera automática.*
- *En la actualidad la principal tendencia en la práctica de la programación es la metodología orientada a objetos (POO), y C# la soporta en su totalidad.*

1. CREAR UN PROYECTO CON C#

En este manual utilizaremos el acrónimo "IDE" (Integrated Development Environment) para referirnos al entorno de desarrollo integrado de C#. Al igual que los procesadores de palabras, programas que facilitan la creación de documentos, el IDE cuenta con herramientas para crear (desarrollar) programas. Todas las herramientas que requerimos para dicho propósito están integradas, por lo que podemos realizar la totalidad de nuestro trabajo dentro del IDE en vez de tener que usar otros programas. En otras palabras, el IDE nos proporciona un entorno de programación completo.

1.2 INSTALACIÓN Y CONFIGURACIÓN

Para seguir los ejercicios es preciso que descargue Microsoft Visual C# 2008 Express Edition. Siga las instrucciones de instalación y registre el producto si el proceso te pide hacerlo.

A continuación, le explicaremos cómo agrupar los programas que cree con C# en una carpeta específica.

Cuando se crea un programa, C# genera varios archivos y los coloca en una nueva carpeta. Al conjunto de archivos resultante se le conoce como "proyecto". Es conveniente crear una carpeta de nivel superior para guardar todos sus proyectos. La carpeta predeterminada que utiliza C# se llama Projects y se encuentra dentro de la carpeta Mis documentos\Visual Studio 2008 en la unidad C. Casi todos los usuarios encuentran adecuada esta opción, pero si por algún motivo usted necesita modificar esa ubicación predeterminada, ponga en práctica estos pasos:

1. *Haga clic en el menú Herramientas, ubicado en la parte superior de la pantalla. Seleccione Opciones… Asegúrese de que la opción Mostrar todas las configuraciones esté seleccionada.*

2. *Haga clic en la opción Proyectos y soluciones y después en el botón … que se encuentra a la derecha de la sección Ubicación de proyectos.*

3. *A continuación, se abrirá la ventana Ubicación del proyecto, en la cual podrá elegir una carpeta existente o crear una nueva. Haga clic en el botón Aceptar cuando termine.*

De aquí en adelante C# guardará todo el trabajo que usted realice en la carpeta de proyectos predeterminada o en la que haya seleccionado de manera específica. En cualquier caso, la primera vez que guarde un proyecto aparecerá un cuadro de diálogo con el botón Examinar, por si necesita utilizar una carpeta distinta a la seleccionada.

1.3 CREANDO EL PRIMER PROGRAMA

El programa que crearemos a continuación desplegará el mensaje Hola mundo en la pantalla de su computadora. Independientemente de lo anterior, deberá seguir estos pasos generales para crear cualquier programa.

Figura 2.1 La página de inicio

Figura 2.2 La ventana de Nuevo proyecto

- *Abra el IDE. Aparecerá la Página de inicio, como se muestra en la Figura 2.1.*
- *Haga clic en el vínculo Crear: Proyecto… En la Figura 2.2 se muestra la ventana Nuevo proyecto que aparecerá en respuesta.*
- *Asegúrese de que esté seleccionada la plantilla Aplicación de Windows Forms. Elija un nombre para su proyecto, mismo que se convertirá también en la identificación de una carpeta. Le recomendamos utilizar sólo letras, dígitos y espacios. En nuestro caso utilizamos el nombre Primer Hola. Haga clic en Aceptar. Aparecerá un área de diseño similar (aunque no necesariamente idéntica) a la que se muestra en la Figura 2.4.*
- *Para facilitar su incursión en el lenguaje C#, es conveniente que el cuadro de herramientas esté siempre visible. Haga clic en el menú Ver y seleccione Cuadro de herramientas. Ahora haga clic en el símbolo de "chincheta" del cuadro de herramientas, como se muestra en la Figura 2.3; el cuadro de herramientas quedará fijo y abierto de manera permanente. Haga clic en Todos los formularios Windows Forms para ver la lista de herramientas disponibles. Ahora su pantalla se verá casi idéntica a la de la Figura 2.4.*

Como se mencionó previamente, los pasos anteriores son comunes en la realización de cualquier proyecto. Realicemos ahora una tarea específica para este proyecto en particular.

Observe el formulario y el cuadro de herramientas ilustrados en la Figura 2.4. A continuación seleccionaremos un control del cuadro de herramientas, y lo colocaremos en el formulario. Éstos son los pasos a seguir:

- *Localice el cuadro de herramientas y haga clic en el control Label.*
- *Desplace el puntero del ratón hacia el formulario. Haga clic y, sin soltar el botón, arrastre para crear una etiqueta como en la Figura 2.5.*
- *Ahora estableceremos algunas propiedades de la etiqueta: haga clic con el botón derecho del ratón sobre la etiqueta y seleccione Propiedades. Desplace el ratón hacia la lista Propiedades en la parte inferior derecha de la pantalla, como se muestra en la Figura 2.6.*

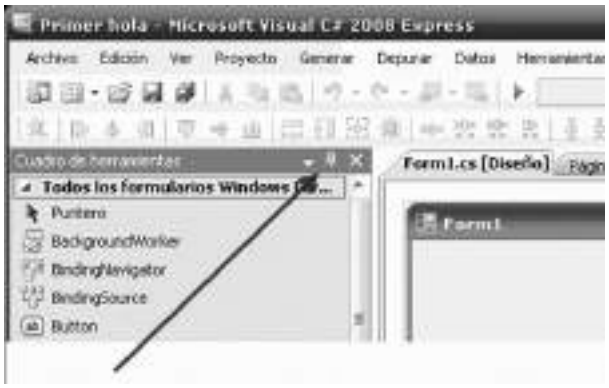


Figura 2.3 El cuadro de herramientas queda fijo y abierto.

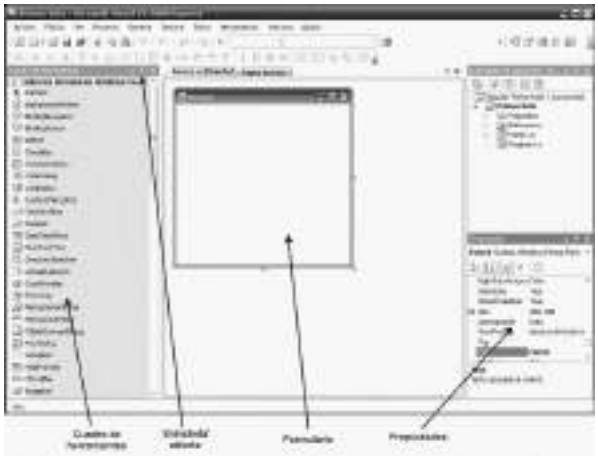


Figura 2.4 El IDE en tiempo de diseño.

Figura 2.5 Se agregó una etiqueta al formulario.

Figura 2.6 Propiedades de la etiqueta.

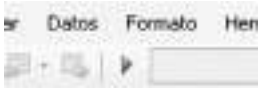


Figura 2.7 Haga clic en la flecha para ejecutar el programa.

- *Desplácese hasta la propiedad Text y sustituya el texto label1 por Hola Mundo.*
- *Ejecutemos ahora el programa, haciendo clic en la flecha que se encuentra en la parte superior del IDE (Figura 2.7); se trata del botón Iniciar depuración.*

Aparecerá una nueva ventana, como se muestra en la Figura 2.8. Es el programa que usted ha creado. Sólo muestra algo de texto, pero es una verdadera ventana en cuanto a que puede moverla, cambiar su tamaño y cerrarla al hacer clic en el icono X que se encuentra en la esquina superior derecha. Experimente con esta ventana, y después ciérrela.

Para guardar su programa y poder usarlo más adelante:

- *Vaya al menú Archivo y seleccione la opción Guardar todo (en adelante, para indicar este tipo de acciones utilizaremos una forma abreviada como ésta: Archivo | Guardar todo).*

- *A continuación, aparecerá la ventana Guardar proyecto, como se muestra en la Figura 2.9. Asegúrese de que la opción Crear directorio para la solución no esté marcada. Deje las demás opciones como están y haga clic en Guardar. Cuando vuelva a guardar el proyecto se utilizarán de manera automática las mismas opciones y ya no aparecerá la ventana Guardar proyecto.*

Ahora puede utilizar el comando Archivo | Salir para cerrar el IDE.

La próxima vez que utilice C# el nombre de su proyecto aparecerá en la Página de inicio, de manera que podrá abrirlo con un solo clic, sin necesidad de repetir el trabajo que hicimos para configurar el proyecto.

Figura 2.8 El programa Primer hola en ejecución.

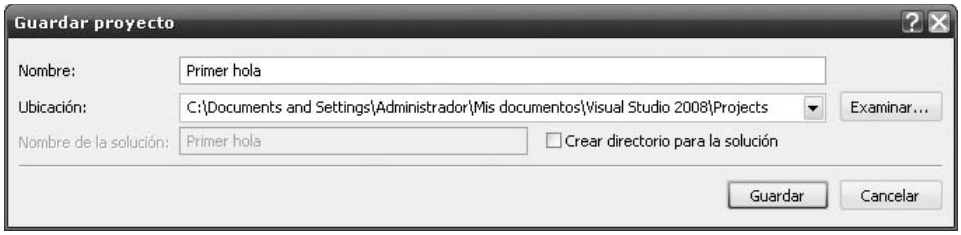


Figura 2.9 La ventana Guardar proyecto, con la opción Crear directorio desactivada.

1.4 LOS CONTROLES EN TIEMPO DE DISEÑO

En nuestro programa Primer Hola colocamos una etiqueta en un formulario y modificamos el texto a desplegar. El propósito principal del ejercicio consistió en repasar los pasos involucrados en la creación de un proyecto. A continuación, conoceremos algunos de los fundamentos de los controles y las propiedades.

¿Qué es un control? Es un "dispositivo" que aparece en la pantalla, ya sea para mostrar información, para permitir que el usuario interactúe con la aplicación, o ambas cosas. El mismo IDE de C# emplea muchos controles. Por ejemplo, en el ejercicio anterior usted utilizó los menús desplegables para guardar proyectos, y el botón Aceptar para confirmar acciones. En el caso de las aplicaciones para Windows, el cuadro de herramientas contiene cerca de 70 controles; parte de la tarea de programación implica seleccionar los controles apropiados,

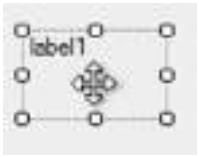
situarlos en un formulario y establecer sus propiedades. A diferencia del "tiempo de ejecución", esta fase tiene relación con el uso de los controles y se denomina "tiempo de diseño". Cuando el programa se ejecuta, el usuario interactúa con los controles. La labor del programador consiste en crear una interfaz gráfica de usuario (GUI) para facilitar dicha interacción. Veamos ahora cómo se pueden manipular los controles en tiempo de diseño.

- *Es posible seleccionar un control del cuadro de herramientas y colocarlo en un formulario. La posición inicial que se le asigne no es importante, ya que podemos modificarla con facilidad.*

- *También es posible mover el control. Si coloca el puntero del ratón sobre un control, a su lado aparecerá una flecha con cuatro puntas, como se muestra en la Figura 2.10. En ese momento podrá arrastrar el control con el ratón.*

- *Se puede cambiar el tamaño del control. Al hacer clic en un control aparecen varios cuadros pequeños (controladores de tamaño) en sus bordes. Coloque el ratón sobre uno de estos cuadros pequeños y aparecerá una flecha con dos puntas, como se muestra en la Figura 2.11. Arrastre el borde o esquina para modificar el tamaño del rectángulo.*

De hecho, el método para cambiar el tamaño depende del control en sí. Por ejemplo, el control tipo etiqueta tiene una propiedad llamada AutoSize (determinación automática de tamaño, de acuerdo con las dimensiones del texto introducido) que establecemos como



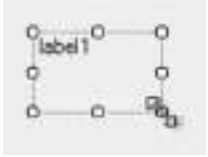


Figura 2.10 Los controles pueden moverse … *Figura 2.11 … y redimensionarse según convenga.*

falsa (False), como se ilustra en la Figura 2.11. Si dejamos la propiedad AutoSize como verdadera (True), el ancho de la etiqueta se determinará con base al ancho del texto que vaya a mostrar, y la altura se determinará en función del tamaño de la fuente del texto. Otros controles permiten variar el ancho arrastrando los controladores con el ratón, pero no la altura (que se establece de acuerdo con las fuentes). Algunos controles —como los botones— permiten cambiar su tamaño en cualquier dirección. En general el tamaño de los controles depende del uso que se les vaya a dar, por lo que le recomendamos experimentar con ellos. Sin embargo, y dado que las etiquetas son tan comunes, no debemos olvidar su propiedad AutoSize.

A continuación, examinaremos las propiedades. He aquí una analogía: los televisores tienen propiedades tales como el color de la carcasa, el tamaño de la pantalla, el canal que está mostrando en un momento dado, su precio y su marca.

De igual manera, cada control tiene un conjunto de propiedades que pueden ser ajustadas en tiempo de diseño para satisfacer nuestros requerimientos. Más adelante veremos cómo cambiar una propiedad en tiempo de ejecución.

Después de colocar un control en el formulario, para ver sus propiedades hay que hacer clic con el botón derecho sobre él y seleccionar Propiedades; de esta manera se desplegará la ventana de propiedades del control elegido. La columna izquierda contiene los nombres de las propiedades, y la columna derecha su valor actual. Para cambiar una propiedad debemos modificar el valor en la columna derecha. En algunas propiedades tal vez sea necesario seleccionar varias opciones adicionales, como al cambiar la configuración de los colores y las fuentes de texto. Algunas veces, cuando el rango de valores a elegir es muy amplio, se requiere abrir una ventana adicional.

Otro de los aspectos vitales de un control es su nombre. Éste no es en sí mismo una propiedad, pero por conveniencia se muestra en la lista de propiedades como (Name). Los paréntesis indican que en realidad no es una propiedad.

Cuando colocamos varias etiquetas en un formulario, el IDE selecciona sus nombres de la siguiente manera:

etiqueta1 etiqueta2 etiqueta3 ...

Estos nombres son aceptables por ahora, pero en los siguientes capítulos veremos que es mejor cambiar los nombres predeterminados de algunos controles por nombres más representativos. Para cambiar el nombre de un control es preciso modificar el texto que está a la derecha de (Name) en la lista de propiedades.

1.5 LOS EVENTOS Y EL CONTROL BUTTON

El programa que creamos en el ejercicio anterior no es muy representativo, dado que siempre muestra las mismas palabras y el usuario no puede interactuar con él. Enseguida enriqueceremos este programa de manera que aparezca un mensaje de texto cuando el usuario haga clic en un botón. Éste es un ejemplo de cómo usar un evento.

Casi todos los eventos son puestos en acción por el usuario, y se generan cuando éste manipula un control de alguna forma en tiempo de ejecución. Cada control tiene varios eventos a los que puede responder, como el clic de un botón del ratón, un doble clic o la colocación del puntero del ratón sobre el control. Otros tipos de eventos no tienen su origen en el usuario; por ejemplo, la notificación de que una página Web ha terminado de descargarse.

En el programa que crearemos a continuación detectaremos un evento (el clic de un botón); después haremos que se despliegue un mensaje de texto en una etiqueta. He aquí la forma en que crearemos la interfaz de usuario:

- *Cree un nuevo proyecto llamado Botón hola.*

- *Coloque una etiqueta y un botón en el formulario. La posición en que lo haga no es importante.*

- *Escriba Haga clic aquí en la propiedad Text del botón.*

- *Modifique la propiedad Text de la etiqueta, de forma que aparezca sin contenido.*

Ejecute el programa, aunque todavía no está completo. Observe que puede hacer clic en el botón, y que éste cambia su aspecto ligeramente para confirmar que está siendo oprimido; no ocurre nada más. Cierre el formulario.

Veamos ahora cómo detectar el evento del clic. Haga doble clic en el botón dentro del formulario de diseño. A continuación, se abrirá un nuevo panel de información, como el que se muestra en la Figura 2.12. Observe las fichas de la parte superior:

Form1.cs Página de inicio Form1.cs [Diseño]

Usted puede hacer clic en esas fichas para cambiar de panel. El panel Form1.cs muestra un programa de C#. A esto se le conoce como "texto del programa", o "código" de C#. Modifiquemos este código utilizando el editor del IDE.

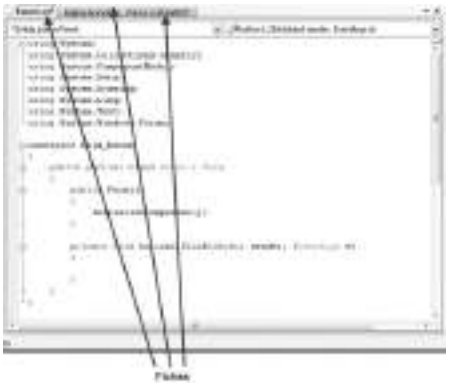


Figura 2.12 Código de C# en el panel de edición.

Desplácese por el código hasta que encuentre la siguiente sección:

```
private void button1_Click(object sender, EventArgs e)
{
}
```

A esta sección de código se le conoce como método. El nombre del método es button1_Click. Cuando el usuario haga clic sobre el botón button1 en tiempo de ejecución, se llevará a cabo, o "ejecutará", cualquier instrucción que coloquemos entre las dos líneas anteriores.

En este programa específico utilizaremos una instrucción para colocar el mensaje Hola mundo en el texto de la etiqueta label1. La instrucción para realizar esta acción es:

label1.Text = "Hola mundo";

Escriba la instrucción exactamente como se muestra aquí; hágalo entre las líneas { y }. En los siguientes capítulos explicaremos el significado exacto de líneas como la anterior (que implican una "instrucción de asignación").

El siguiente paso es ejecutar el programa. Para ello hay dos posibilidades:

- *Si el programa se ejecuta correctamente, al hacer clic en el botón observará que aparece el mensaje Hola mundo en la etiqueta.*

Figura 2.13 Acción errónea. Marque la casilla de verificación y haga clic en No.

- *La otra posibilidad es que el programa no se ejecute debido a un error. En este caso C# mostrará un mensaje como el de la Figura 2.13. Marque la casilla de verificación de la opción No volver a mostrar este cuadro de diálogo, y haga clic en No para continuar. El mensaje no volverá a aparecer; en lo sucesivo la única evidencia del error será un subrayado en el texto del código.*

De todas formas, hay que corregir el error: revise el código, corrija cualesquiera errores de escritura, y ejecute el programa de nuevo. Más adelante hablaremos sobre los errores con más detalle.

He aquí las nuevas características de este programa:

- *Puede responder al clic en un botón. Al proceso de colocar código de manera que se lleve a cabo la acción correspondiente cuando ocurra un evento se le conoce como "manejar" el evento.*
- *Modifica el valor de la propiedad de un control en tiempo de ejecución. Anteriormente sólo hacíamos esto en tiempo de diseño. Ésta es una parte muy importante de la programación, ya que a menudo tenemos que mostrar resultados colocándolos en cierta propiedad de un control.*

1.6 APERTURA DE UN PROYECTO EXISTENTE

Para volver a abrir un proyecto creado con anterioridad, guarde y cierre el proyecto en progreso y vaya a la Página de inicio. En ella se muestran los proyectos en los que se ha trabajado más recientemente; para abrir uno basta con hacer clic en su nombre. Si el

proyecto que busca no se encuentra en la lista, haga clic en el vínculo Abrir: Proyecto… y navegue hasta la carpeta apropiada. Dentro de la carpeta hay un archivo de tipo . sln (solución). Seleccione este archivo y abra el proyecto. Para ver el formulario vaya al Explorador de soluciones, ubicado en la parte superior derecha de la ventana, y haga doble clic en el nombre del formulario.

1.7 DOCUMENTACIÓN DE LOS VALORES DE LAS PROPIEDADES

En este libro necesitamos proveerle los valores de las propiedades que emplearemos en los controles. Cuando las propiedades sean pocas podremos explicarlas en unos cuantos enunciados, pero si la cantidad es mayor utilizaremos una tabla. En general tenemos varios controles; cada control tiene varias propiedades y cada propiedad tiene un valor. Tenga en cuenta que hay una gran cantidad de pro- piedades para cada control, pero sólo listaremos aquellas que usted tenga que modificar. Las demás mantendrán su valor predeterminado. He aquí un ejemplo:

Control	*Propiedad*	*Valor*
label1	*Text*	*Hola mundo*
label1	*BackColor*	*Red*
button1	*Text*	*Haga clic aquí*

Los valores listados corresponden a los siguientes controles:

- *label1 cuyo texto es Hola mundo, y cuyo color de fondo es rojo;*
- *button1 cuyo texto es Haga clic aquí.*

Tenga cuidado al escribir los nombres de las propiedades. Hablaremos sobre esto con más detalle en capítulos posteriores; por ahora basta decir que debe hacerlo con letra mayúscula inicial, y sin espacios.

1.8 ERRORES EN LOS PROGRAMAS

Es común que ocurran errores en los programas; por fortuna, el IDE puede detectar algunos por nosotros. He aquí un ejemplo de error:

label1.Tixt = "Hola mundo"

Si usted escribe esta línea y ejecuta el programa, el IDE la subrayará. Al colocar el puntero del ratón sobre la parte subrayada aparecerá una explicación del error. Tenga en cuenta que

la explicación podría ser difícil de comprender (incluso para un experto), o que tal vez el error se encuentre en otra parte del programa. Sin embargo, el primer paso deberá consistir siempre en inspeccionar el área subrayada para ver si hay errores de escritura. A medida que vaya aprendiendo más sobre el lenguaje C# mejorará su habilidad para detectar errores.

Una vez corregidos los errores subrayados hay que ejecutar el programa. De hecho, antes de que esto ocurra se activa otro programa llamado compilador, el cual realiza comprobaciones en el código para detectar, quizá, más errores. Sólo hasta que se hayan corregido todos los errores de compilación su programa podrá ejecutarse.

Los errores de compilación se muestran en la ventana Lista de errores, la cual se abre debajo del código. Al hacer doble clic en el error dentro de la ventana de resultados, el IDE nos llevará al punto del código donde se debe corregir el error.

Cualquiera que empiece a escribir programas enfrentará muchos errores de compilación. No se desanime; esto es parte del proceso de aprendizaje.

Tras corregir los errores de compilación el programa podrá ejecutarse. En este punto podríamos notar que lo hace de manera incorrecta, lo cual querría decir que hay un "error en tiempo de ejecución", o bug. Dichos errores son más difíciles de corregir, y en consecuencia es necesario realizar una depuración.

1.9 FUNCIONES DEL EDITOR

El editor no se limita a mostrar lo que escribe el programador; las siguientes son algunas de sus virtudes adicionales:

- *Ofrece las operaciones estándar de cortar, copiar y pegar en el portapapeles, gracias a lo cual usted podrá copiar código de otros programas de Windows.*

- *Para escribir programas más grandes se requieren controles y eventos adicionales. Las listas desplegables que están en la parte superior del editor nos permiten seleccionar controles, además de un evento específico para dicho control. El editor se coloca de manera automática en el método apropiado.*

- *El editor desplegará el código de C# que se va tecleando de acuerdo con ciertas reglas. Por ejemplo, se insertarán espacios de manera automática alrededor del símbolo =.*

- *Algunas líneas necesitarán sangrías —desplazamiento del texto a la derecha—, para lo cual hay que insertar espacios adicionales en blanco. Por lo general esto se hace en intervalos de cuatro espacios. Como veremos en capítulos posteriores, el código de C# consiste de secciones dentro de otras secciones, y las sangrías nos ayudan a indicar en dónde inicia y termina cada sección.*

Al proceso de aplicar sangrías a un programa también se le conoce como darle formato. El IDE puede hacerlo de manera automática a través del menú:

Edición | Avanzado | Dar formato al documento Es conveniente dar formato al código con frecuencia.

- *Cada tipo de control cuenta con un conjunto de propiedades particular. Si usted escribe el nombre de un control seguido de un punto, como se muestra a continuación:*

label1.

y espera un poco, el editor le proporcionará una lista de las propiedades de la etiqueta label1. Si esta herramienta no se encuentra configurada en su sistema, puede activarla seleccionando la opción:

Herramientas | Opciones | Editor de texto | C#

y marcando después la casilla de verificación Lista de miembros automática.

- *Las secciones del código en las que no estemos trabajando pueden contraerse en una sola línea de código. Para ello hay que hacer clic en el pequeño símbolo "–" que está a la izquierda del editor. Para expandir una sección haga clic en el símbolo "+". Encontrará estos símbolos al lado de líneas como la siguiente:*

private void button1_Click (object sender, EventArgs e)

No es preciso que usted recuerde todas estas herramientas, pues siempre están al alcance de su mano para ayudarlo.

Una de las tareas relacionadas con la distribución del código que no se realizan de manera automática es la división de líneas extensas. Para asegurar que toda una línea esté visible en la pantalla podemos elegir un lugar adecuado para dividirla y oprimir la tecla Enter. El IDE aplicará una sangría de cuatro espacios a la segunda parte de la línea. He aquí un ejemplo:

private void button1_Click (object sender, EventArgs e)

Si presionamos la tecla Enter después de la coma, el código aparecerá así:

private void button1_Click (object sender, EventArgs e)

Muchas veces el IDE crea líneas muy extensas que, por supuesto, no deben contener errores, de manera que es mejor no dividirlas. Sin embargo, a medida que sea más experimentado en su manejo de C#, algunas de estas largas líneas serán de su propia creación, y resultará muy conveniente dividirlas de manera que pueda verse todo el texto a la vez. Con esto también se mejora la legibilidad del código impreso (conocido como listado).

1.10 EL CUADRO DE MENSAJES

Anteriormente utilizamos el control tipo etiqueta para mostrar texto en la pantalla, pero también podemos usar un cuadro de mensajes para lograrlo. Este control no aparece en el cuadro de herramientas, debido a que no ocupa un espacio permanente en un formulario; sólo aparece cuando es requerido. El siguiente es un fragmento de código que muestra el mensaje Hola mundo dentro de un cuadro de mensajes al hacer clic en un botón:

private void button1_Click(object sender, EventArgs e)

{

MessageBox.Show("Hola mundo");

}

La Figura 2.14 muestra lo que ocurre al ejecutar el programa y hacer clic en el botón: se despliega el cuadro de mensajes, y debemos hacer clic en el botón Aceptar para que desaparezca. Esta característica implica que los cuadros de mensajes se utilizan para mostrar mensajes importantes que el usuario no puede ignorar.

Para utilizar un cuadro de mensajes escriba una línea como la anterior, pero utilice su propio mensaje dentro de las comillas dobles. En este momento no explicaremos el propósito de la instrucción Show ni por qué se requieren los paréntesis. Hablaremos sobre todo esto cuando estudiemos los métodos con más detalle.

Figura 2.14 Un cuadro de mensajes

1.11 AYUDA

El sistema de ayuda funciona con base en dos posibles fuentes: una local, ubicada en su propia computadora, y la otra a través de Internet. La fuente de Internet se actualiza con frecuencia; sin embargo, la versión local es suficiente cuando se empieza a trabajar con C#: En caso de que usted no cuente con una conexión a Internet, el programa utilizará automáticamente la ayuda local.

Si desea especificar de dónde debe provenir la ayuda, puede reconfigurar el sistema de la siguiente forma:

1. *En la ventana principal de C#, seleccione Ayuda | Buscar...*
2. *En la nueva ventana de ayuda seleccione Ayuda | Ayuda sobre la Ayuda. A continuación, aparecerán varias opciones sobre cuál fuente de ayuda tendrá mayor prioridad.*

El sistema de ayuda de C# es extenso, pero si usted es nuevo en materia de programación tal vez la información que contiene le resulte complicada. Es más provechoso cuando se tiene algo de experiencia con C# y se requiere un detalle técnico preciso.

Las opciones más útiles del menú Ayuda son Índice y Buscar, ya que nos permiten escribir cierto texto para que el sistema localice las páginas que puedan servirnos. La diferencia es

que Índice sólo busca en los títulos de las páginas, mientras que Buscar también examina el contenido de las mismas.

1.11.1 Fundamentos de programación

- *Los controles pueden colocarse en un formulario en tiempo de diseño.*
- *Es posible establecer las propiedades de los controles en tiempo de diseño.*
- *Los programas son capaces de modificar las propiedades en tiempo de ejecución.*
- *Cuando ocurre un evento (como hacer clic en un botón), el sistema de C# utiliza el método apropiado, pero es el programador quien debe colocar código dentro del método para manejar ese evento.*

1.11.2 Errores comunes de programación

- *Olvidar dar por terminada la ejecución de su programa antes de tratar de modificar el formulario o el código.*
- *Confundir el formulario en tiempo de diseño con el formulario en tiempo de ejecución.*

1.11.3 Secretos de codificación

- *En el código de C#, para referirnos a la propiedad de un control utilizamos su nombre seguido por un punto y por el nombre de la propiedad, como en:*

label1.Text

- *A una sección de código entre:*

```
private void botonNumero_Click(Object sender, EventArgs e)
{
}
```

se le conoce como método.

- *Los cuadros de mensaje no se colocan directamente en los formularios. Para hacer que aparezcan en pantalla debemos usar la siguiente instrucción:*

```
MessageBox.Show ("Aquí va el texto que usted desee");
```

1.11.4 Nuevos elementos del lenguaje

Una introducción a las propiedades, métodos y eventos.

1.11.5 Nuevas características del IDE

- *Los programas están contenidos en proyectos.*
- *El IDE crea una carpeta que contiene los archivos necesarios para un proyecto.*
- *Es posible mover y cambiar el tamaño de los controles en los formularios.*
- *El cuadro de herramientas contiene una amplia diversidad de controles.*
- *Al hacer clic con el botón derecho del ratón sobre un control podemos seleccionar sus propiedades.*
- *Al hacer doble clic en un botón en tiempo de diseño se crean métodos para manejar eventos.*

1.11.6 Resumen

Parte de la tarea de programación implica colocar controles en formularios y establecer sus propiedades iniciales. El IDE de C# realiza esta tarea directamente, pero es necesario que practique con el IDE y lea sobre él.

y renderizado Interacción con el usuario - Entrada

Para permitir al usuario interactuar con nuestras aplicaciones, Unity proporciona automáticamente una clase "Input".

Para más información en información véase: https://docs.unity3d.com/Manual/class-InputManager.html https://docs.unity3d.com/ScriptReference/Input.html

La entrada ofrece la posibilidad de verificar si se pulsa un botón físico (joypad, ratón, teclado) y cuándo se pulsa, de obtener la posición de uno o varios ejes físicos (ej. joystick analógico, desplazamiento del ratón) o de crear ejes virtuales a partir de dos botones digitales (ej. velocidad - botón de ralentización).

Unity es capaz de recibir datos de pantallas táctiles y de diferentes sensores del smartphone (por ejemplo, giroscopio, cámara y acelerómetro).

```
void Actualizar(){
        if(Input.GetKeyUp(KeyCode.Space)){          //Entrada desde teclado o
                joypad Debug.Log("Se ha liberado el espacio");
        }
}
```

```
void Actualizar(){
        float myVerticalValue = Input.GetAxis("Vertical");      //entrada desde
joystick analógico o ejes virtuales
         if(miValorVertical != 0)
        {
                //eg mover al jugador
        }
}
```

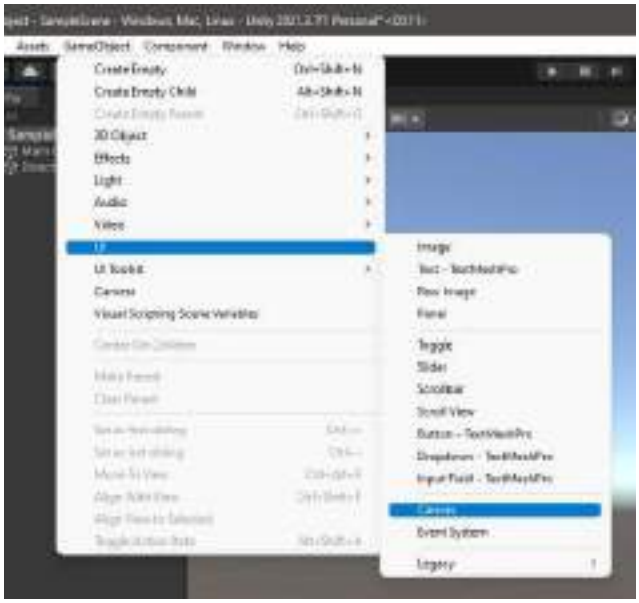
```
void Actualizar(){
        if(Input.GetMouseButton(0)){        //para entrada de ratón y
                touchscren Debug.Log("Se ha pulsado el botón 0 del
                ratón");
```

```
                Rayo myRay = Camera.main.ScreenPointToRay(Input.mousePosition);
                 if(Physics.Raycast(myRay)){
                                //Hice clic en algo porque miRayo golpeó algo
                 }
        }
}

void Actualizar(){
        Vector3 myDeviceAcceleration = Input.acceleration; //Entrada del acelerómetro
                if(myDeviceAcceleration.magnitude > 10)
        {
         Debug.Log("El dispositivo se ha apagado antes");
         }
}
```

Es bueno comprobar y gestionar la entrada en la función Actualizar (porque la función se invoca cada vez que se crea un fotograma) para tener una capacidad de respuesta proporcional a la velocidad de fotogramas de la aplicación.

Interacción con el usuario - Salida

En caso de salida de registro (como texto) en la consola es posible utilizar la función:
`Debug.Log("customText");`
La mayor parte de la interacción entre el usuario y la aplicación tiene lugar a través de la interfaz de usuario o UI.
Unity le permite crear fácilmente una interfaz de usuario utilizando el GameObject Canvas. Para más información:
https://docs.unity3d.com/2021.3/Documentation/Manual/UICanvas.html
https://docs.unity3d.com/ScriptReference/Canvas.html



Canvas permite crear una interfaz de usuario programable y personalizable utilizando diferentes elementos.

Es posible añadir texto y botones.

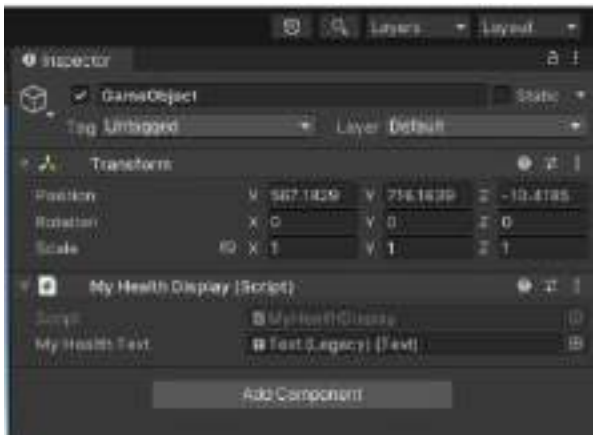
Para crear un código que controle el texto a través del botón, añada un componente script a un GameObject vacío.

```
usando UnityEngine.UI
public class PantallaSalud : MonoComportamiento
        { public TextoMiSalud;
        int miSalud = 10;
        public void myButtonFunction()                    {//función para conectar
al botón
```

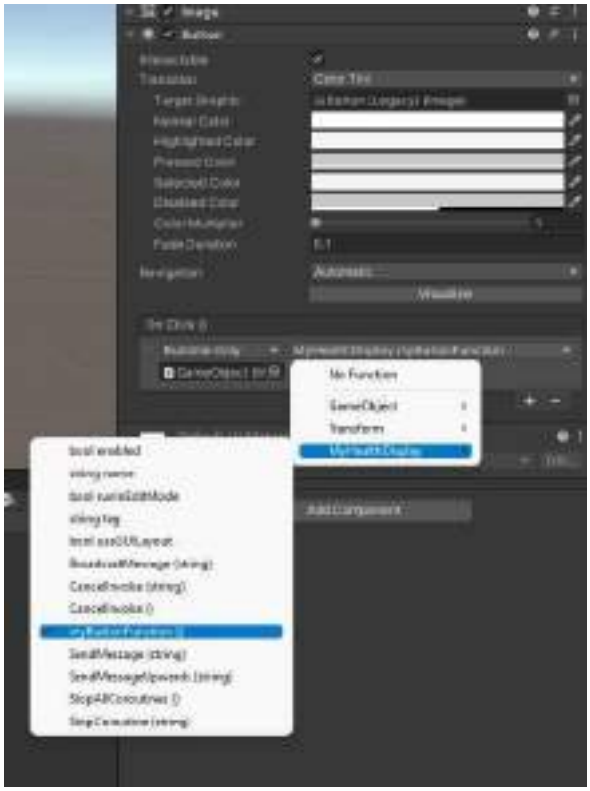
```
        miSalud--;                                  //lower la healt
        miSaludTexto.texto = "SALUD : " + miSalud;          //actualiza la etiqueta
    }
```

}

Vincule el texto a la secuencia de comandos mediante el atributo Texto público (Mi texto de salud).



Enlaza el botón al script Empty GameObject a través de la función onclick() del botón usando la función pública (void myButtonFunction()).



Física - Colisión

Las colisiones en Unity son procesadas gracias a los componentes Collider(BoxCollider, SphereCollider, MeshCollider) y gestionado mediante eventos eventos (OnCollisionEnter, OnCollisionExit, OnCollisionStay). Para más información:
https://docs.unity3d.com/Manual/collision- section.html
https://docs.unity3d.com/ScriptReference/Collision.html

```
void OnCollisionEnter(Colisión acolisión){
        if(acollision.relativeVelocity.magnitude >= 10)  {//if  a  bola  colisiona
                                                          con gameObject
                Destroy(gameObject);                      //destruir
                gameObject Destroy(acollision.gameObject); //Destruir la
                bola
        }
}
```

En Edición > Configuración del proyecto > Etiquetas y capas es posible añadir nuevas capas

En Edición > Configuración del proyecto > Física > Matriz de colisiones es posible definir entre qué capas deben producirse las colisiones.

La modificación de la transformada del componente o el uso de las funciones "transform.Translate o Transform.Rotate" no producen colisiones.

Para evitar cálculos innecesarios, introduzca el componente de colisión sólo cuando sea necesario.

RigidBody

Los principales aspectos de la dinámica se simulan mediante el componente Rigidbody. https://docs.unity3d.com/Manual/rigidbody-physics-section.html
https://docs.unity3d.com/ScriptReference/Rigidbody.html

Mediante el uso del componente Rigidbody puede aplicar fuerzas y momentos, puede

establecer la velocidad, las fuerzas de fricción y la gravedad y crear movimientos que produzcan colisiones.

```
private void Update(){
    int fuerza de salto =
    100; int velocidad =
    10;
    if(Input.GetKeyDown(KeyCode.Space))
```

```
        gameObject.rigidbody.AddForce(Vector3.up * jumpForce, ForceMode.Impulse);
    if(Input.GetKeyDown(KeyCode.R)
        gameObject.rigidbody.AddTorque(transform.right * speed, ForceMode.Force);

}
```

Es preferible desvincular el uso de simulaciones físicas de la velocidad de fotogramas mediante la función.
FixedUpdate piuttosto che Update.

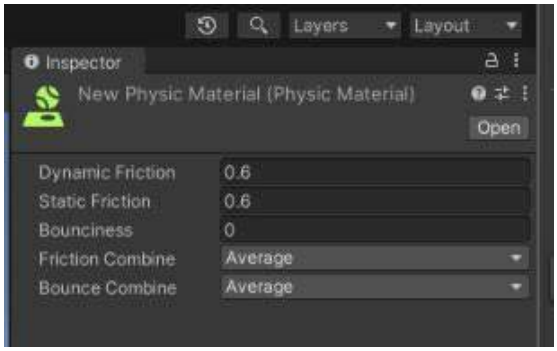
```
private bool goingForward;
private void Actualizar(){
        goingForward = Input.GetKey(KeyCode.UpArrow);
}
private void FixedUpdate(){
        if(avanzando){
                gameObject.rigidbody.AddForce(Vector3.Forward * speed,
ForceMode.Force);
        }
}
```

Las velocidades de movimiento muy rápidas pueden no gestionarse a tiempo.

Material físico

Los Physic Materials utilizados en un GameObject con un collider permiten definir algunas de sus propiedades físicas como la creación de fricción estática y dinámica y la elasticidad del objeto.

https://docs.unity3d.com/Manual/class-PhysicMaterial.html



Articulaciones

Es posible enlazar dos GameObject (con RigidBody) a través del componente

joint. https://docs.unity3d.com/Manual/joints-section.html

FixedJoint crea una articulación fija entre el objeto al que se ha unido este componente y otro objeto especificado. El resultado es una articulación fija en la que uno de los objetos sigue físicamente los movimientos del otro (igual que un gameObject hijo sigue a un GameObject padre).
Hingejoint crea una articulación de bisagra entre el objeto al que se ha unido este componente y otro objeto. El resultado es una restricción fija entre los dos objetos, que sólo pueden girar sobre el eje de la bisagra.

SpringJoint crea una articulación de muelle entre el objeto al que se ha unido este componente y otro objeto. El resultado es una articulación entre dos objetos que sigue la física de un muelle.

Rayos (obstáculos y balas)

La clase Rayo permite crear rayos para sondear la presencia de objetos a partir de un punto y una dirección determinados.
La clase RaycastHit se utiliza para memorizar las colisiones a través de los rayos.
La función Physics.Raycast( ) se utiliza para comprobar si hay colisiones a lo largo del radio.
https://docs.unity3d.com/ScriptReference/Physics.Raycast.html

```
Rayo miRayo = nuevo Rayo(miRayoPuntoInicio,miRayoDirección);
//Rayo(transform.position, transform.forward) es un rayo que parte del objeto y es dirigido adelante
RaycastHit miHit;
float rayDistance = 100.0;
if (Physics.Raycast(myRay, out myHit, rayDistance )){
   Debug.Log(myHit.collider.name); //podemos obtener el collider del objeto hit y su
```

nombre u otros componentes

}

https://docs.unity3d.com/Manual/CameraRays.html

La clase Física proporciona varias funcionalidades muy útiles para la física.

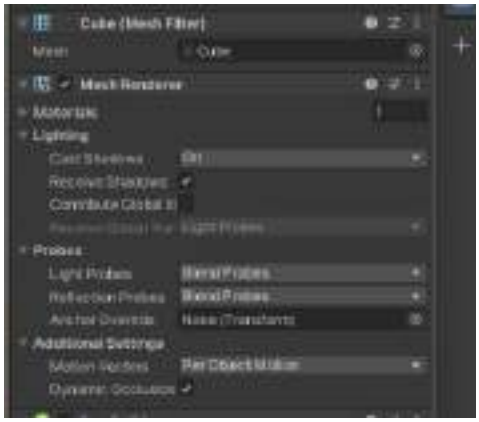
https://docs.unity3d.com/ScriptReference/Physics.html

https://docs.unity3d.com/Manual/PhysicsSection.html

Conceptos básicos de renderizado

Existen diferentes componentes que caracterizan los gráficos de los objetos. El componente filtro de malla define la forma de nuestro GameObject.
https://docs.unity3d.com/Manual/mesh.html

Con el componente Mesh Render es posible definir si las mallas de nuestro GameObject tienen que proyectar y recibir sombras y como deben comportarse cuando reflejan la luz.
https://docs.unity3d.com/Manual/class-MeshRenderer.html



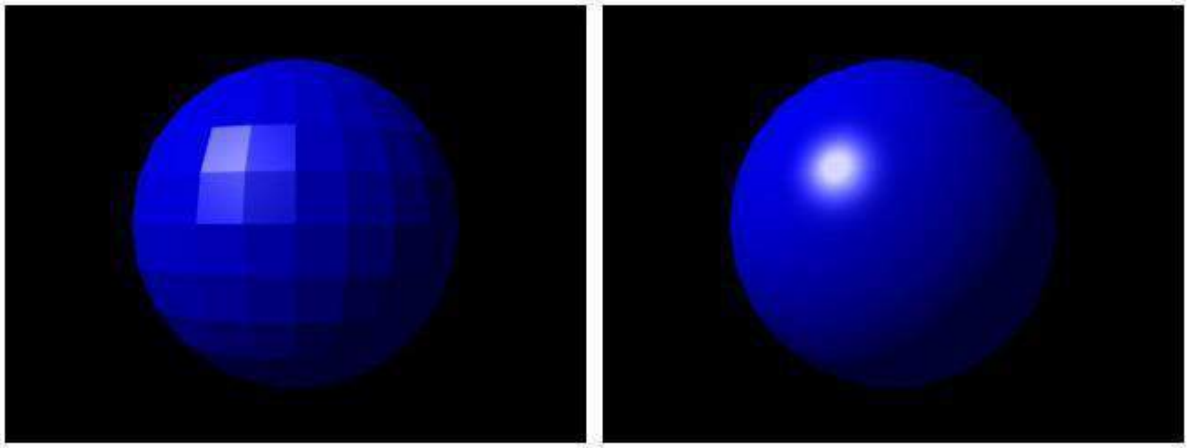
Sombreadores

Los shaders son modelos matemáticos utilizados por Unity para indicar a la tarjeta gráfica cómo representar los objetos en pantalla en función de las luces, las sombras, los efectos...

Cada sombreador realiza cálculos diferentes y consigue efectos visuales distintos (por ejemplo, sombras realistas, efecto dibujo animado, efecto píxel...)
https://docs.unity3d.com/Manual/Shaders.html

Los sombreadores de vértices calculan los reflejos y las sombras en una cara de la malla promediando lo que ocurre en los vértices de la cara (más rápido pero de menor calidad. Los reflejos y las sombras siguen la forma de la malla).
Los sombreadores de píxeles calculan las luces y sombras teniendo en cuenta todos

los aspectos relacionados con un objeto situado en un único píxel de la pantalla (más lentos pero de mayor calidad, las sombras y los reflejos son independientes de la forma de la malla a la que están unidos).



Materiales

Los materiales definen los valores que los shaders deben utilizar durante los cálculos gráficos. Usando el mismo shader es posible definir materiales con diferentes colores, transparencias, brillo, opacidad.

No todos los sombreadores pueden manejar todas las propiedades (por ejemplo, la transparencia).

https://docs.unity3d.com/Manual/Materials.html

Textura

Las texturas pueden definir el color y otras propiedades del material píxel a píxel.

Importar una textura en Unity es tan simple como arrastrarla a sus Assets. Para utilizarla en un material, basta con arrastrarla a la propiedad del material para la que queremos utilizarla. (por ejemplo, albedo).

Las texturas de alta resolución complican y alargan exponencialmente los cálculos gráficos. https://docs.unity3d.com/Manual/Textures.html

Fuentes de luz

En Unity hay diferentes fuentes de luz.

La luz puntual es una luz situada en un punto de la escena que emite luz en todas direcciones por igual.

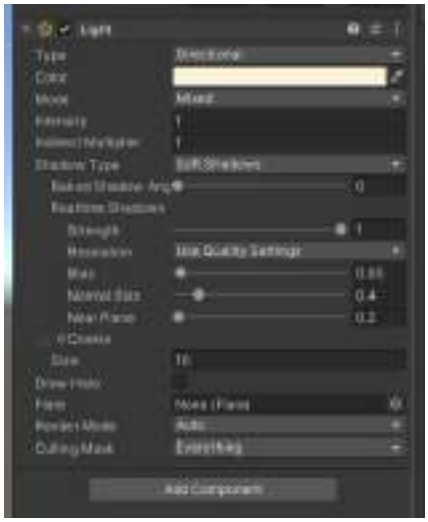
Luz puntual es una luz situada en un punto de la escena que emite luz en forma de cono. Luz direccional es una luz situada a una distancia infinita que emite luz en una sola dirección.

Area Light es una Luz que está definida por un polígono en la Escena, y emite luz en todas las direcciones uniformemente a través de su superficie pero sólo desde un lado.

Para cada fuente luminosa es posible definir su intensidad, su color y, en algunos casos, la dirección, el efecto efecto la distancia y si o no este produce sombras.

https://docs.unity3d.com/Manual/Lighting.html

https://docs.unity3d.com/Manual/LightingOverview.html



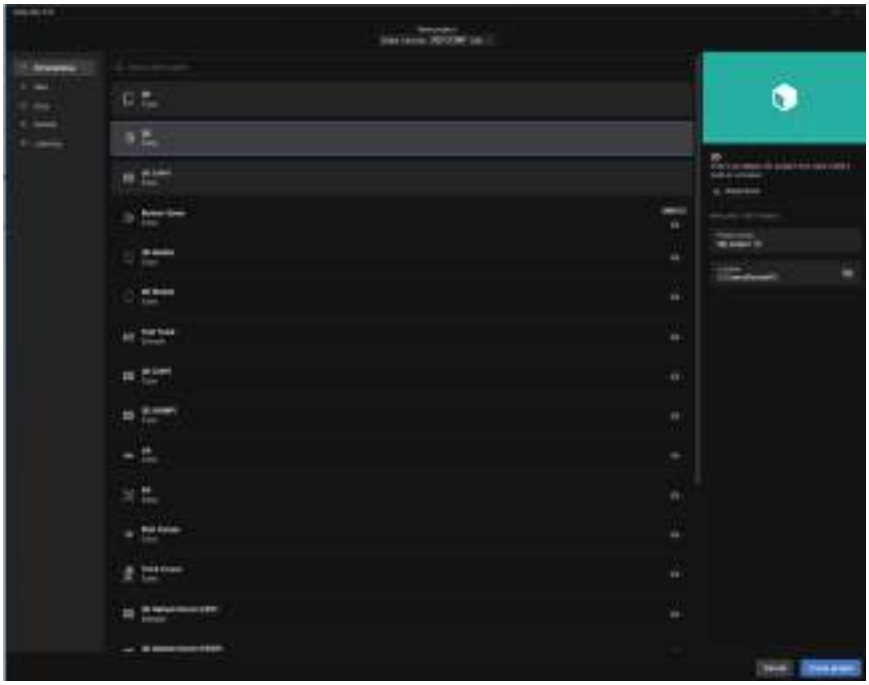
En los dispositivos móviles, no se recomienda el uso de sombras.

Unidad 5 - Guía práctica con ejemplos de código

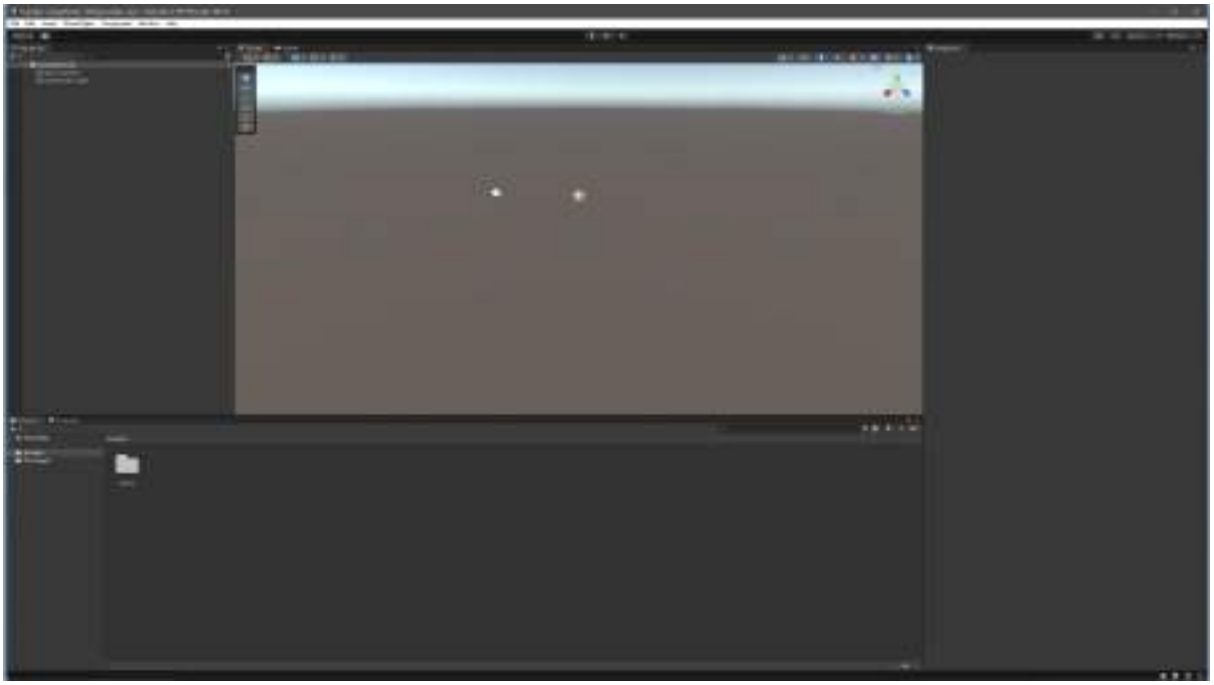
En este capítulo veremos cómo realizar algunas de las funciones del videojuego.

Vamos a crear un nuevo proyecto pulsando el botón *Nuevo proyecto*.

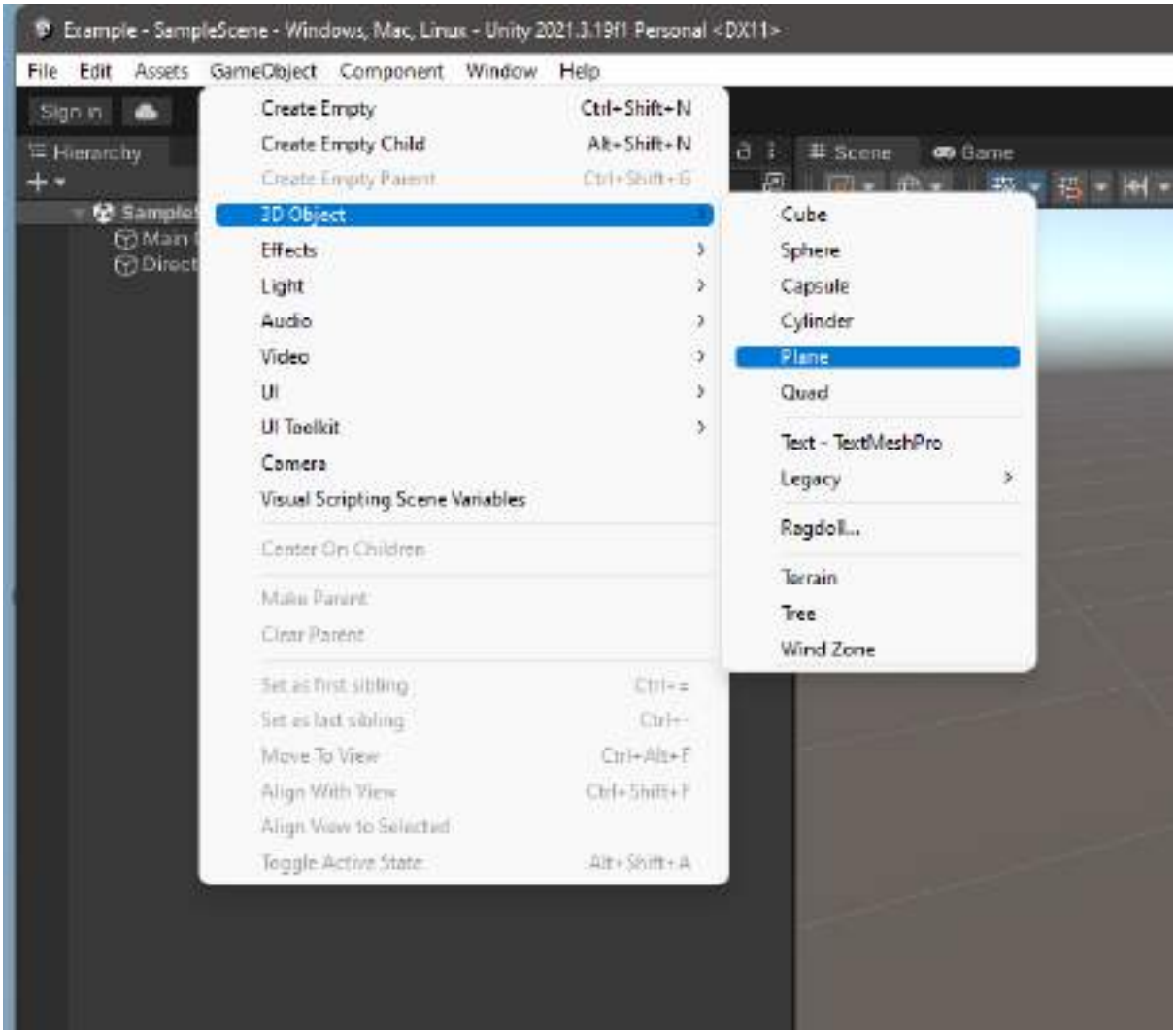
Ahora, selecciona "3D-core", establece el nombre del proyecto y selecciona "Crear proyecto".



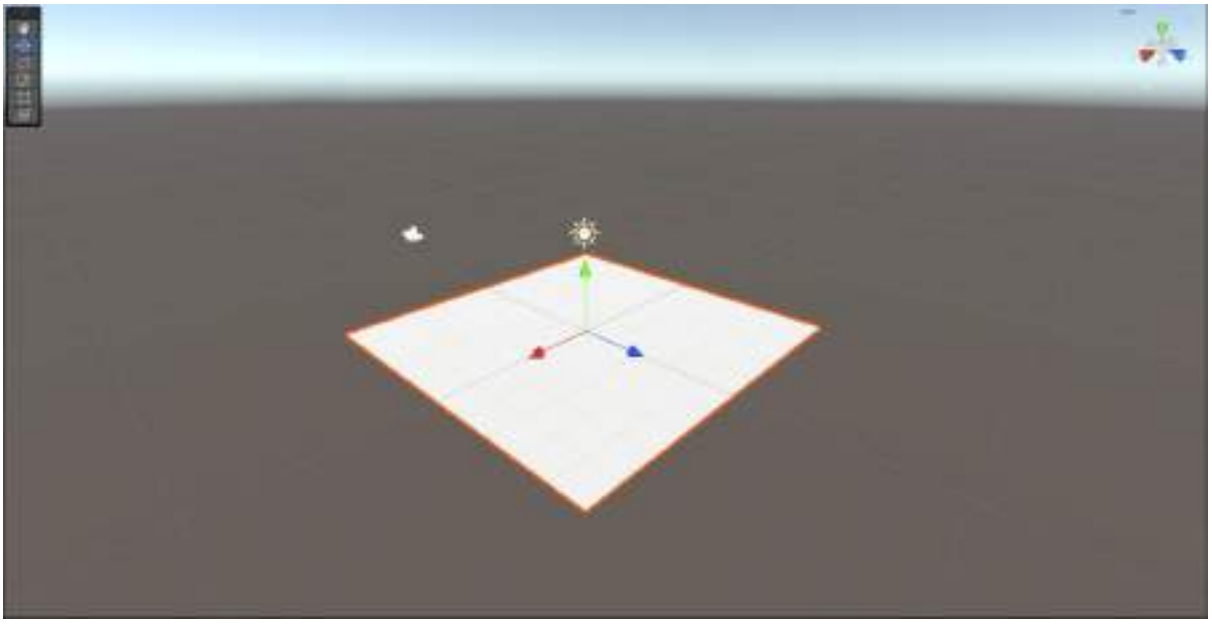
Una vez creado el proyecto, aparecerá la interfaz de usuario de Unity.



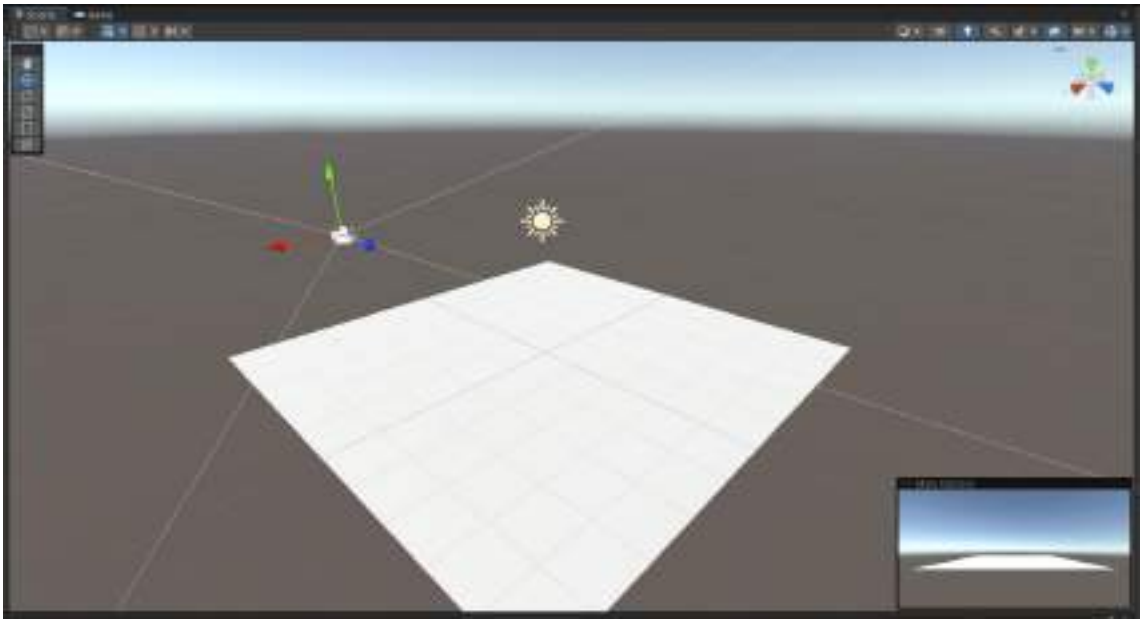
Vamos a añadir un plano desde el menù, siguiendo los pasos escritos a continuación: GameObject -> Objeto 3D -> Plano

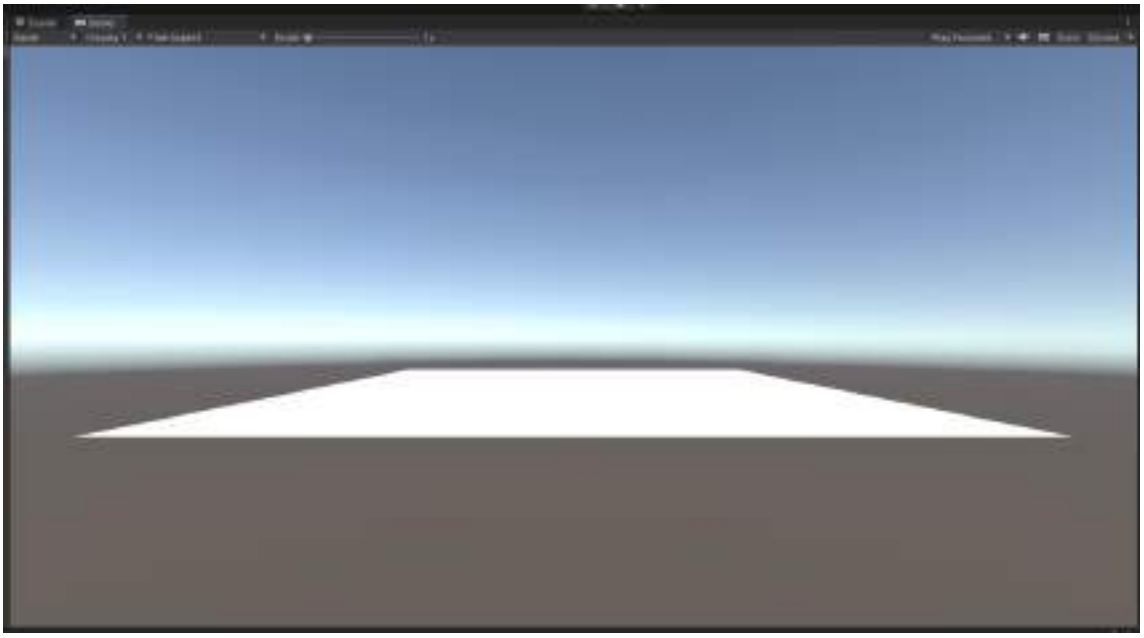


El avión tendrá este aspecto:

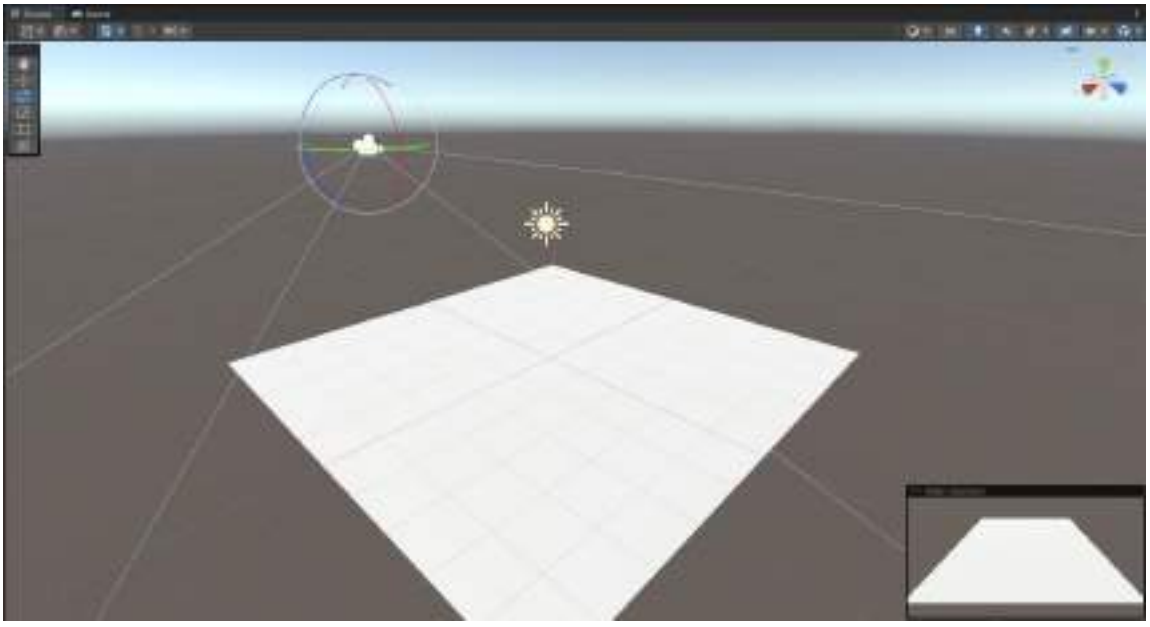


Seleccione la cámara y asegúrese de que el plano está encuadrado correctamente



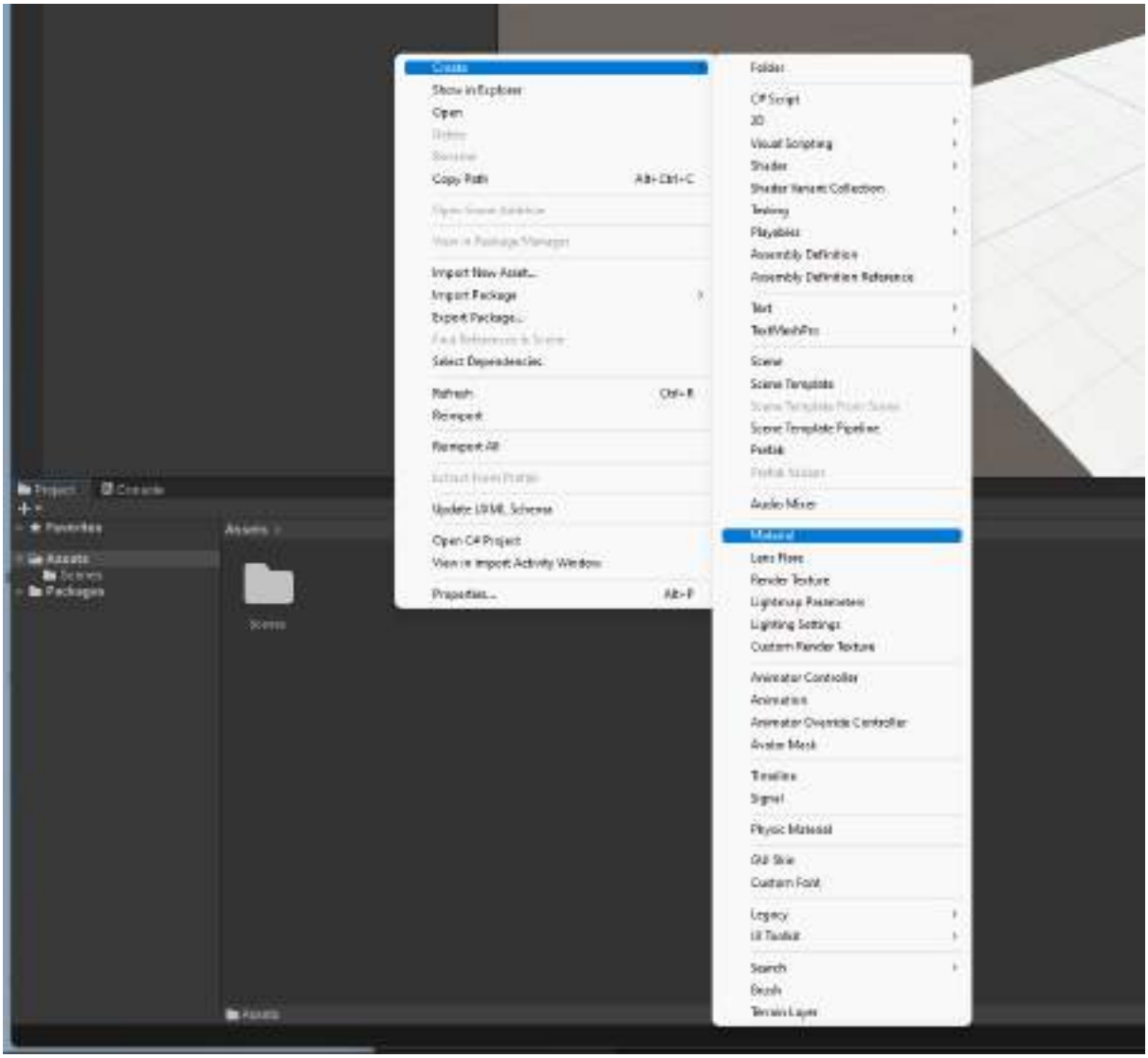


Mueva y gire la cámara con las herramientas adecuadas en la parte izquierda de la vista de escena.



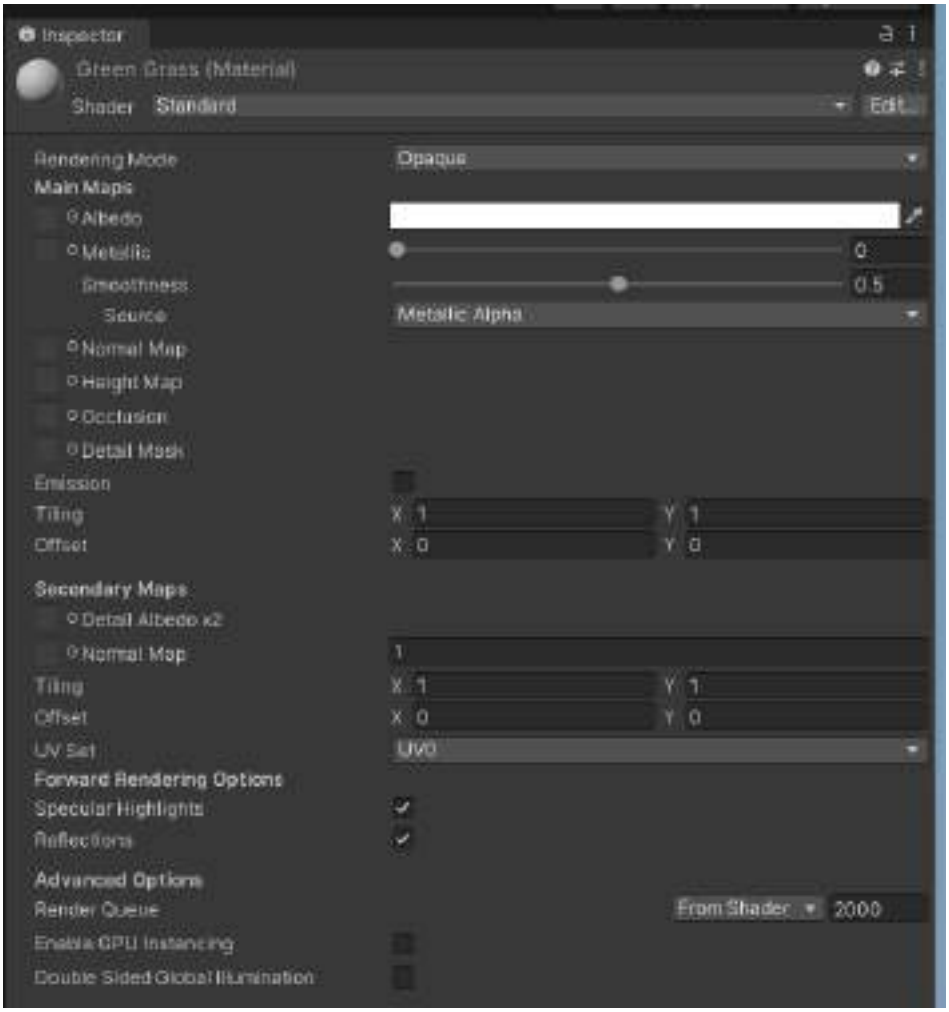
Vamos a crear un nuevo material para colorear el suelo de verde.

Haga clic con el botón izquierdo del ratón en el panel Activos, Crear -> Material.

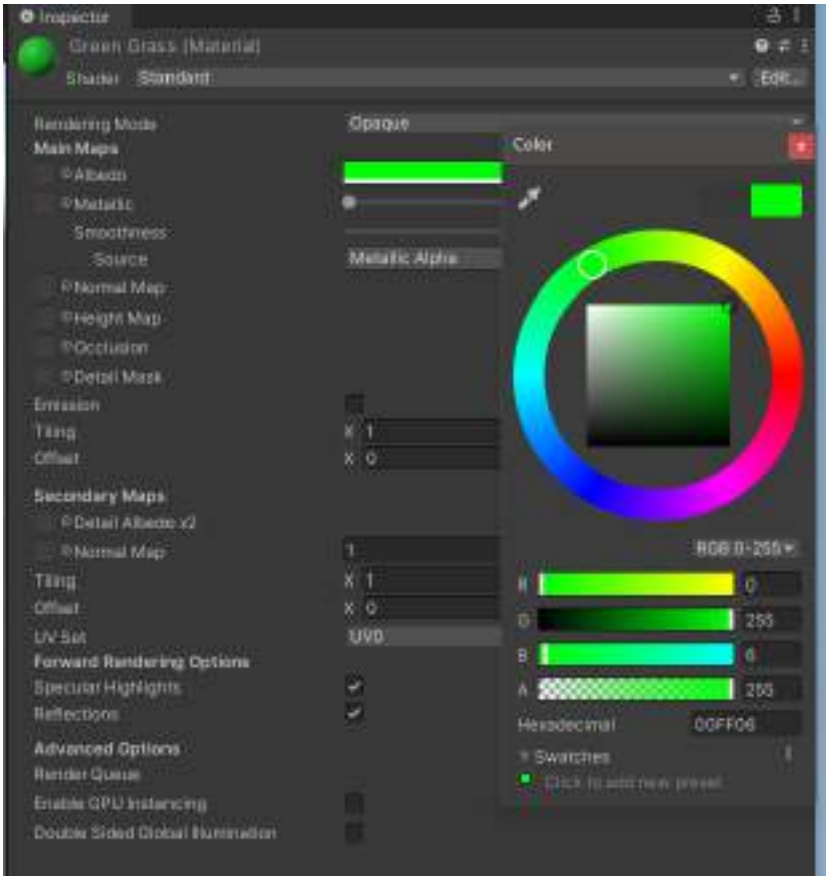


Nombremos el material

Ajuste el color en el panel "inspector" pulsando sobre el rectángulo coloreado en blanco.



Elige el color.



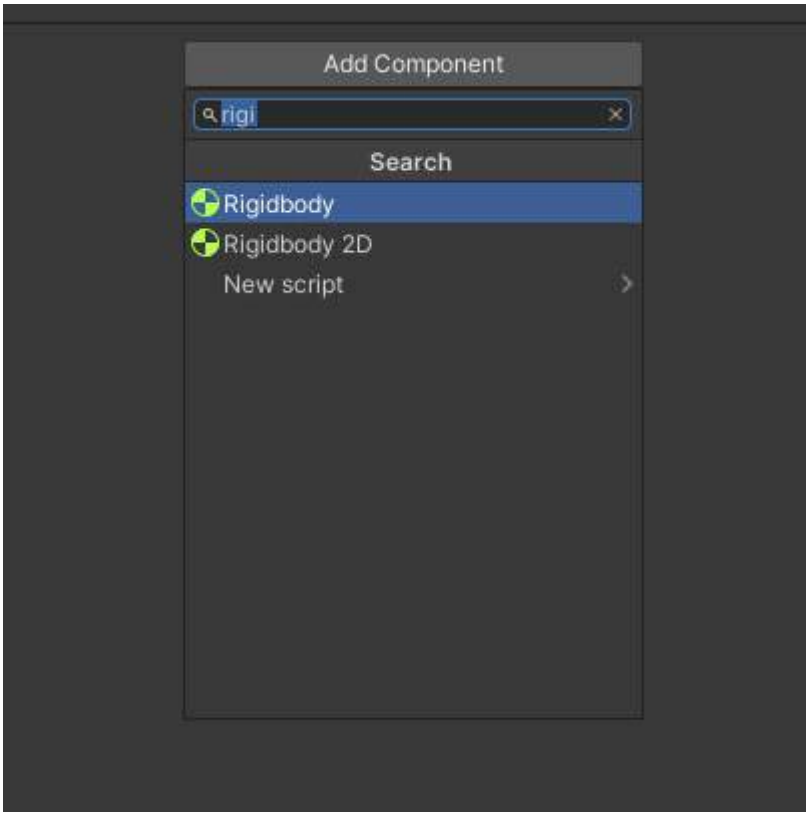
Arrastre el material 

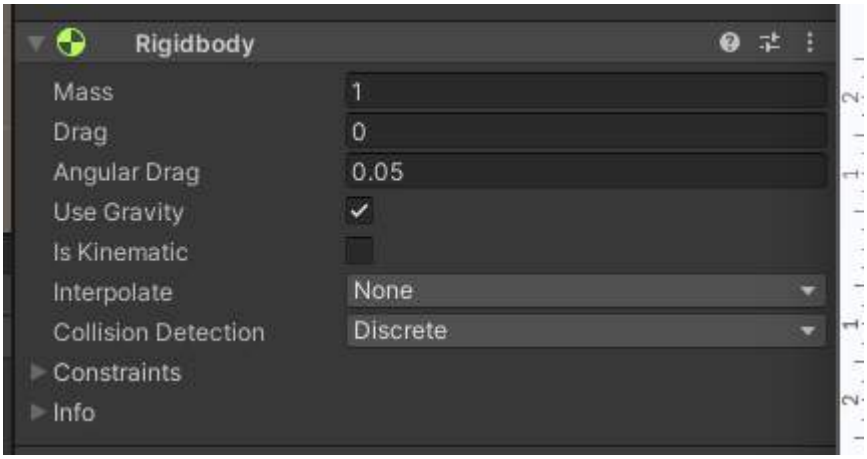
desde los Activos y traiga el plano a la vista de escena para cambiar su material.

Con el mismo método crea un cubo y coloréalo de azul claro.

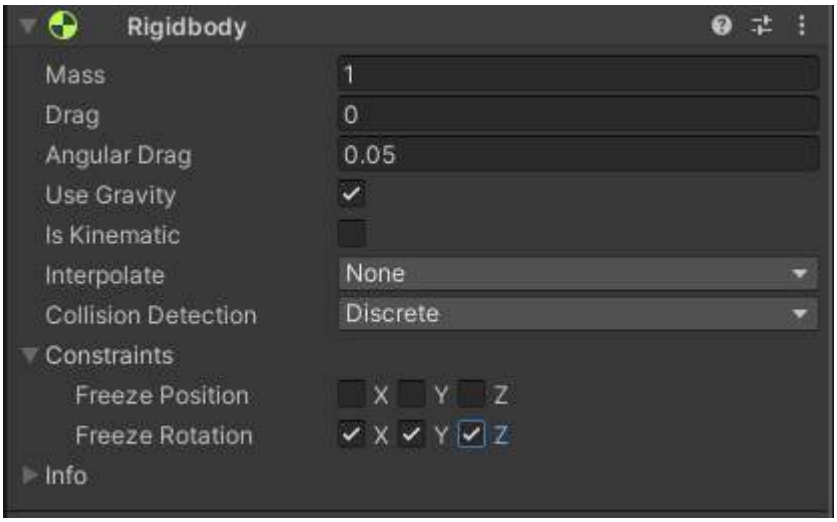
Añadamos el componente Rigidbody:
Selecciona el cubo, pulsa Añadir Componente en el panel Inspector, busca y selecciona "Rigidbody".

Esto permitirá controlar el cubo mediante la física y someterlo a la gravedad.





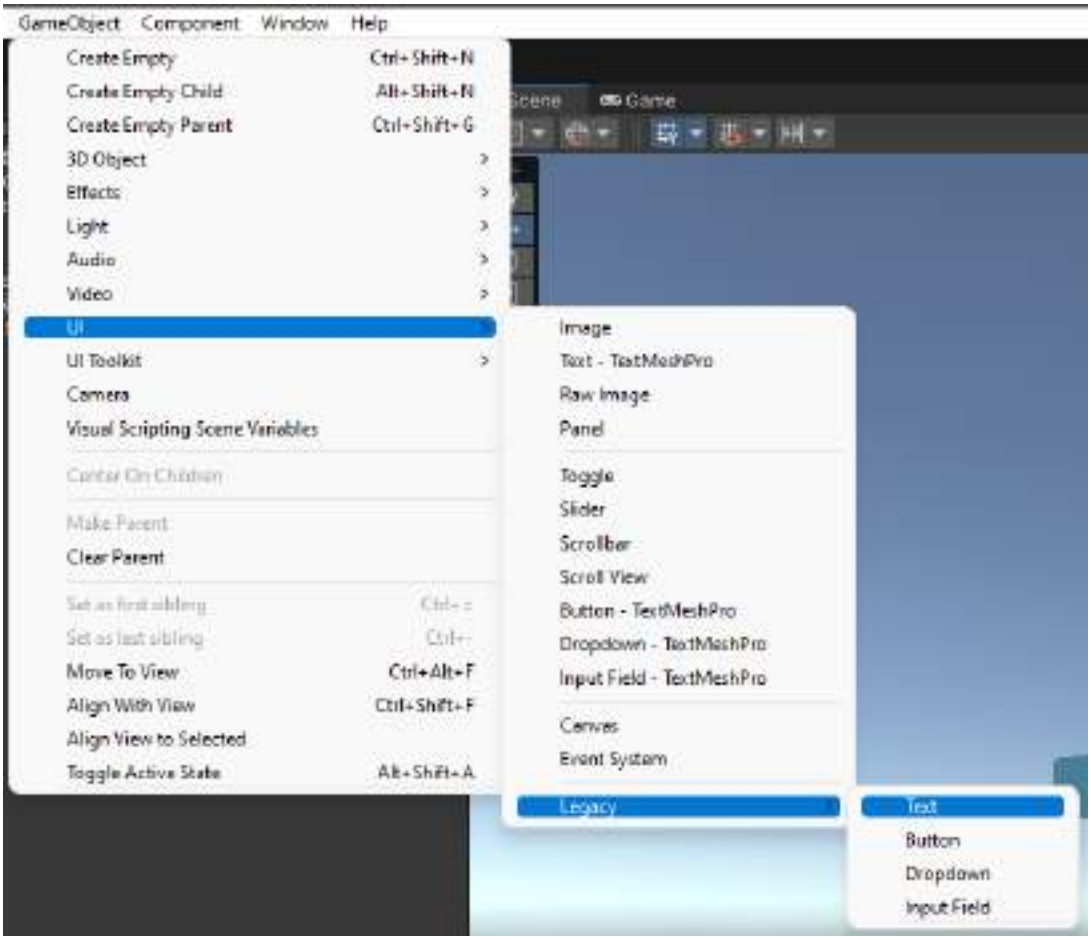
Seleccione Congelar Rotación (X, Y, Z) si desea evitar que el cubo gire mientras se mueve.



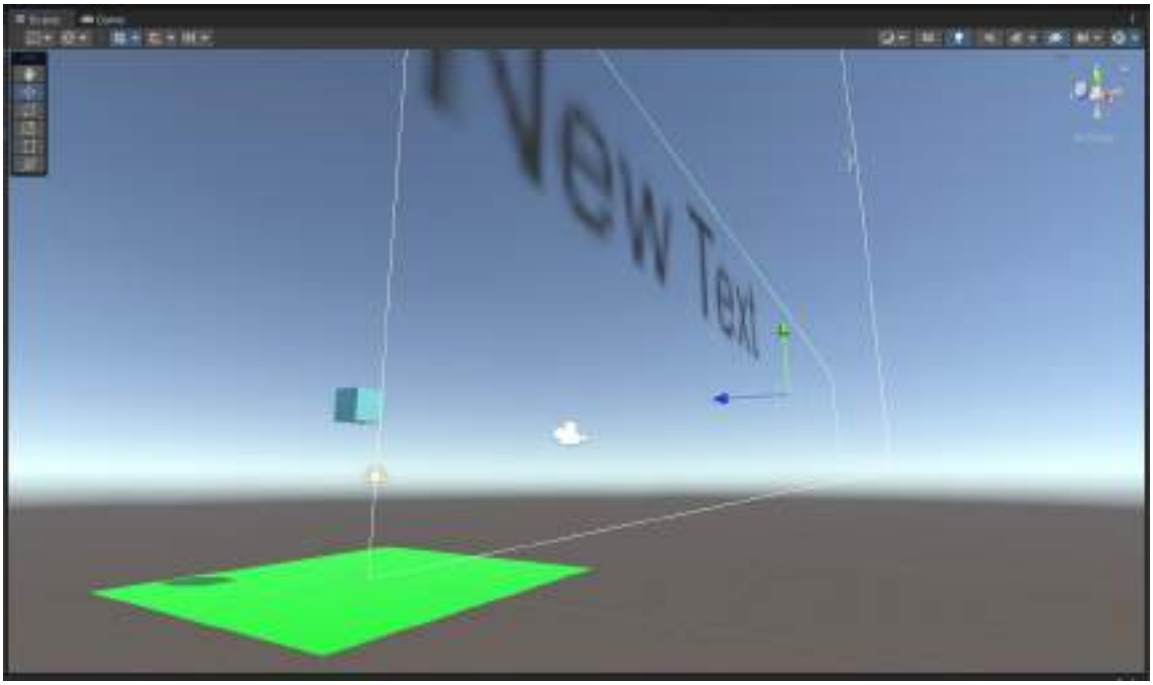
(consejo: en lugar de mantener pulsado el botón izquierdo del ratón y moverte, puedes utilizar WASD que te permite seleccionar un objeto del panel "Jerarquía" y pulsar MAYÚS+F para centrar la vista directamente en ese objeto).

Añadamos el texto superpuesto para comunicarnos con el usuario:

GameObject -> UI -> Legacy -> Text

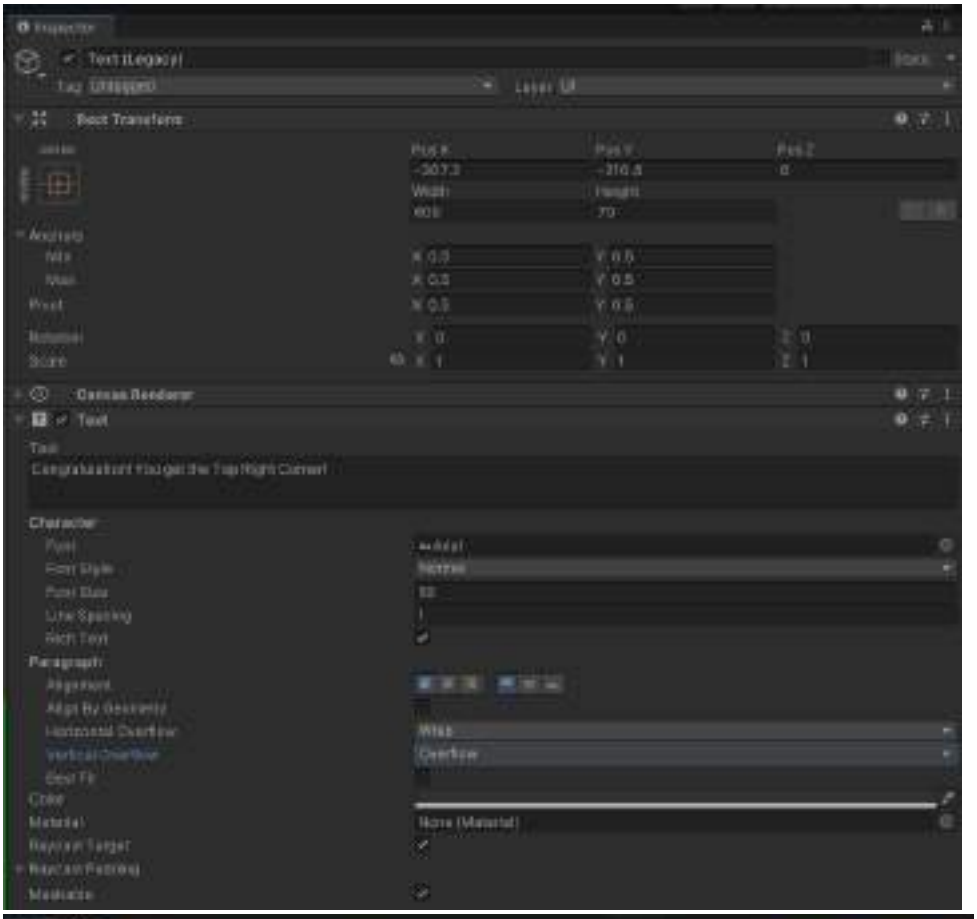


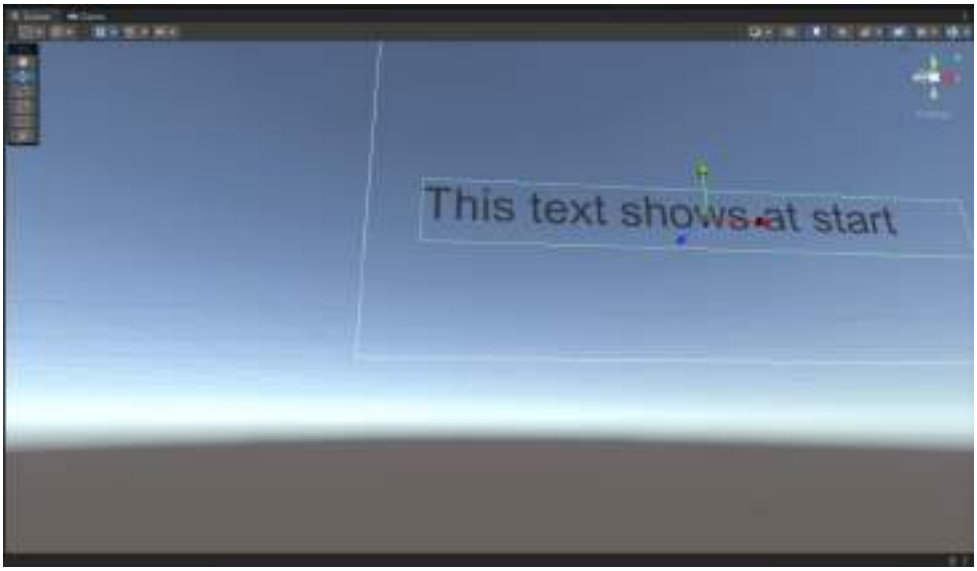
Aparecerá un "Canvas" (Lienzo) muy grande donde podemos añadir textos y otros elementos para la interfaz gráfica de usuario.



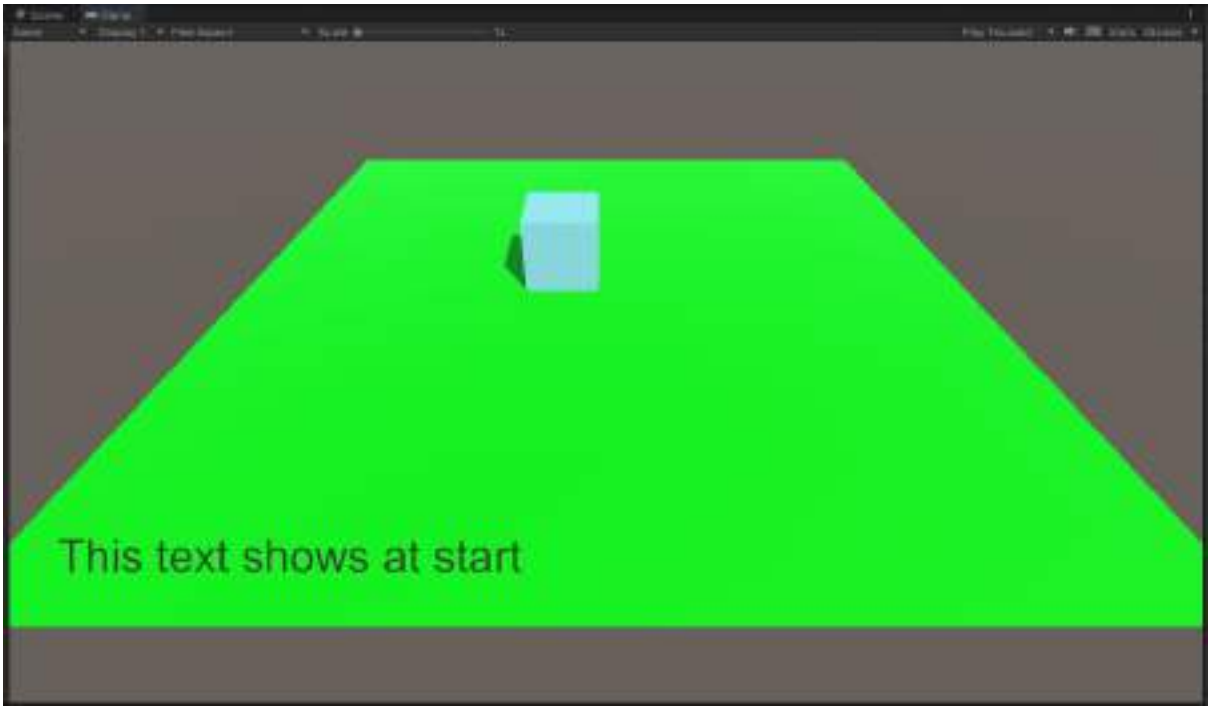
Seleccione Texto (Legacy)
Cambie la anchura, la altura, el texto y el tamaño de la fuente desde el panel Inspector;

Mueva el texto a la posición que desee.





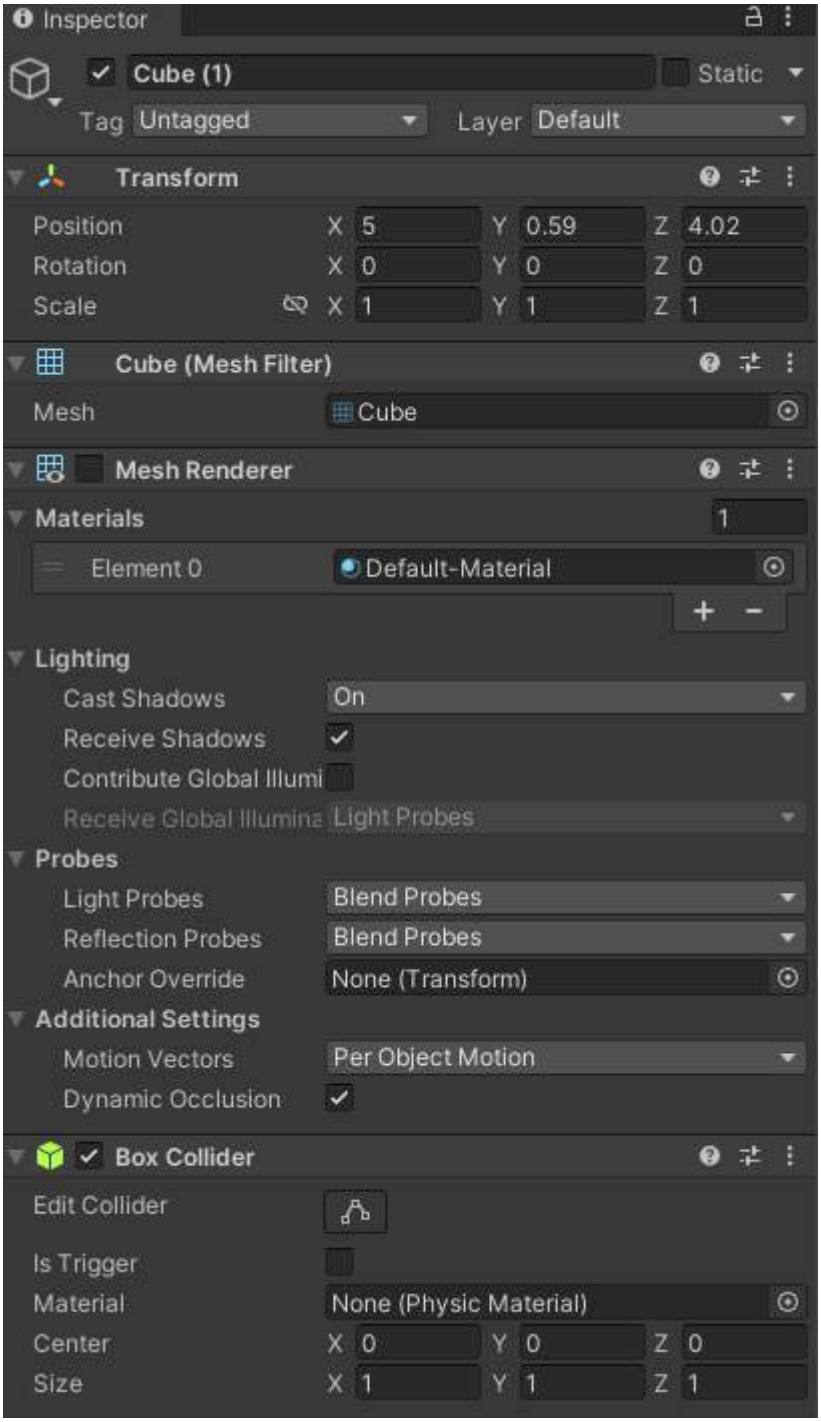
La posición del texto en el lienzo corresponde a la posición en la pantalla durante el juego.



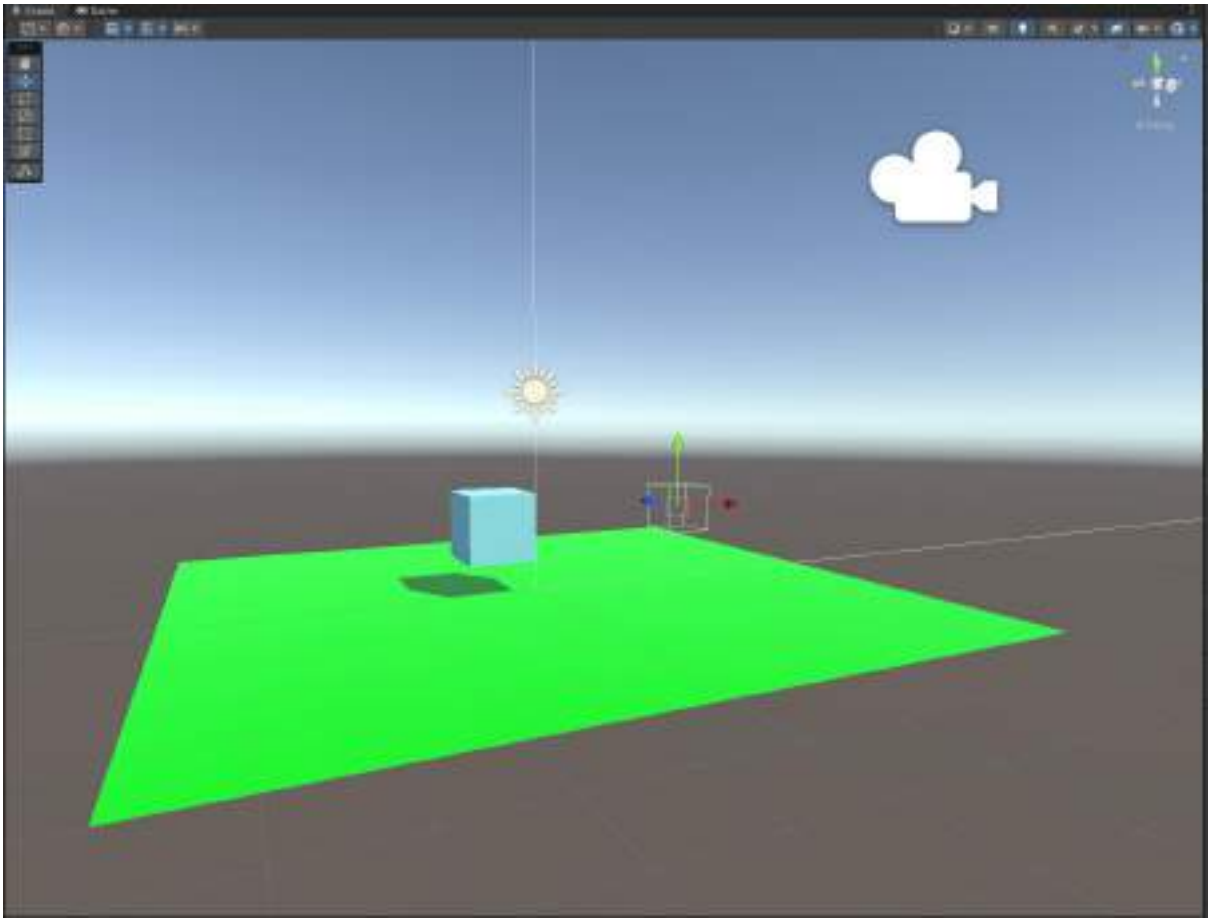
Vamos a crear otro cubo:

En Inspector -> Box Collider marca la casilla "is Trigger" para transformar el cubo en un area que inic
un evento.

Desactive la casilla Renderizado de malla para hacer invisible el área que se ha creado.



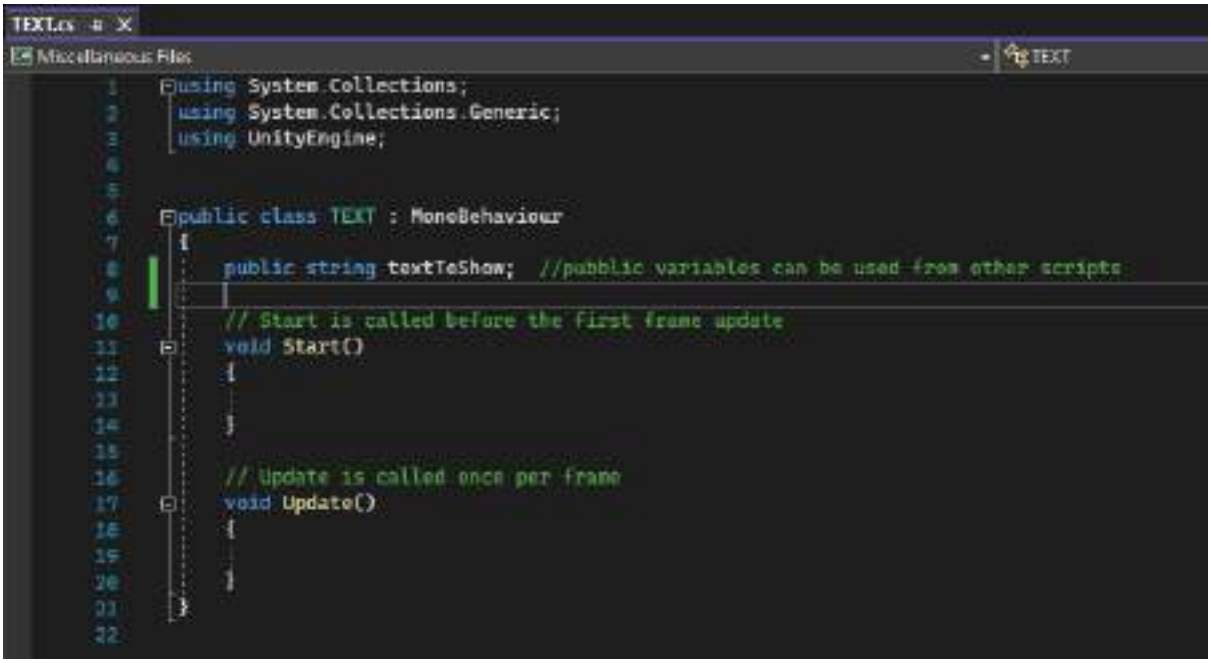
Localiza el nuevo cubo en la esquina superior derecha de la superficie.



Añada un componente de script llamado TEXTO.

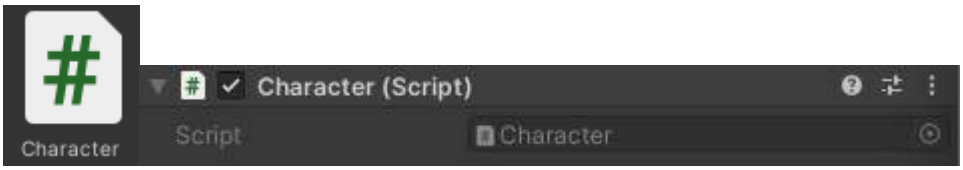
Haga doble clic en el archivo que aparecen en los activos y añada la instrucción de código en la línea 8.


```
Cadena pública textToShow;
```



Guarde y cierre el editor de código.
La instrucción añadida nos permite utilizar y establecer la cadena textToShow desde el panel de inspección.
Cambiar el valor de Texto a mostrar en *"¡Felicidades! Has conseguido la esquina superior derecha!"*

Selecciona el cubo azul claro y añádele también un componente script.
Nombra el script "Personaje" .



Abre el script y añade esta línea de código después del primer corchete antes de void Start()

```
public float forceAmount = 0.8f; //esta riga impone el valor de la fuerza
```

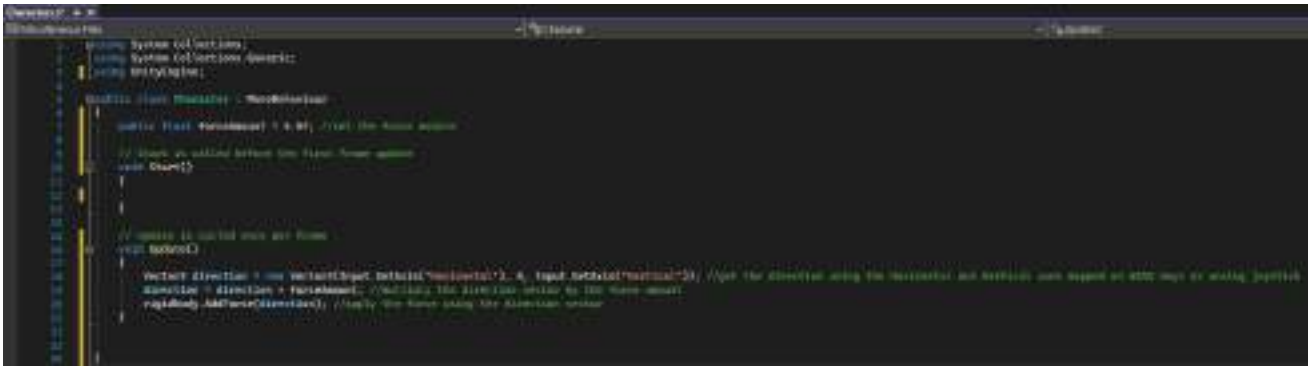
Añada las siguientes líneas de código entre las dos llaves de la función Actualizar.

```
Vector3 direction = new Vector3(Input.GetAxis("Horizontal"), 0, Input.GetAxis("Vertical")); //toma dirección utilizando los botones "Horizontal" y "Vertical" de los pulsadores WASD o de las teclas analógicas.
```

```
        direction=direction*forceAmount; /moltiplica la dirección de la fuerza
        rigidbody.AddForce(direction); //aplica la fuerza utilizando la dirección del vector
```

Estas instrucciones nos permitirán mover nuestro cubo azul claro utilizando las teclas WASD del teclado.

El código será similar al siguiente:



GUARDAR

Habilite la posibilidad de utilizar instrucciones relacionadas con la interfaz de usuario añadiendo lo siguiente en la línea 4

```
usando UnityEngine.UI;
```

Inserte la siguiente cadena después de la declaración de "forceAmount"

```
public TextoACambiar;
```

La línea de código anterior nos permitirá vincular textToChange al texto en el lienzo utilizando el pane Inspector.

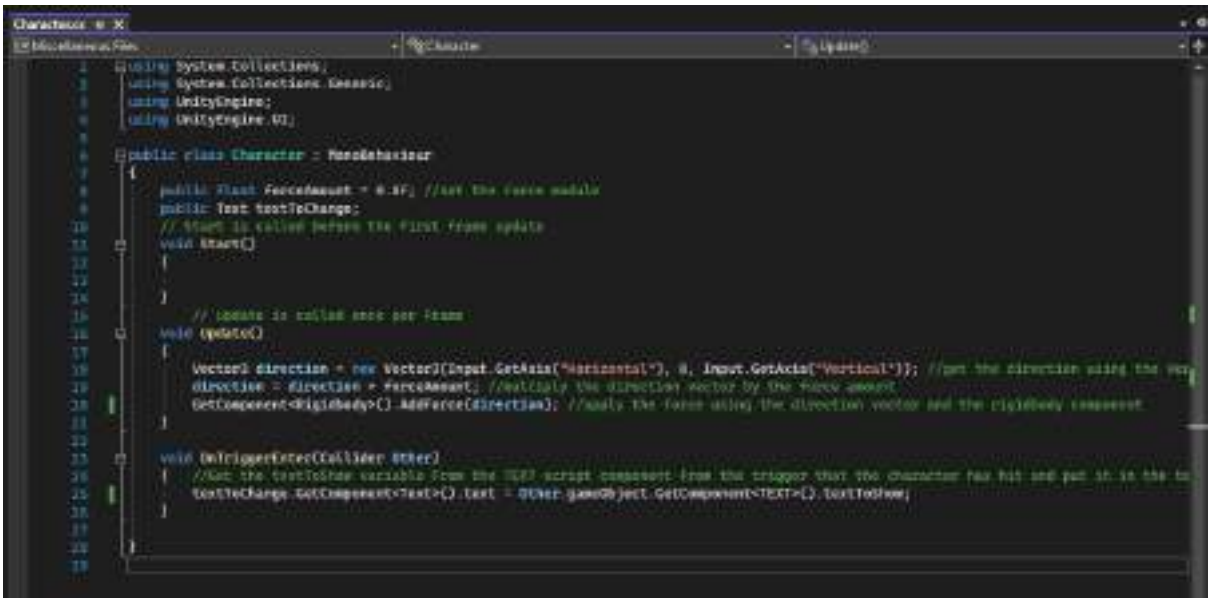
Añade este código antes del cierre de la última llave.

```
void OnTriggerEnter(Collider Otro)
    {
        textToChange.GetComponent<Text>().text =
Other.gameObject.GetComponent<TEXT>().textToShow;
```

}

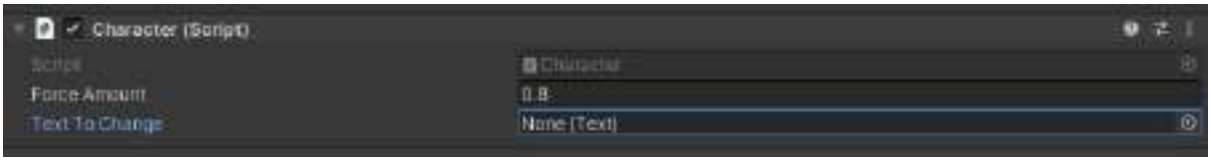
La función anterior cambiará el texto superpuesto por el del código del cubo invisible establecido com área de activación.

El código será similar al siguiente:



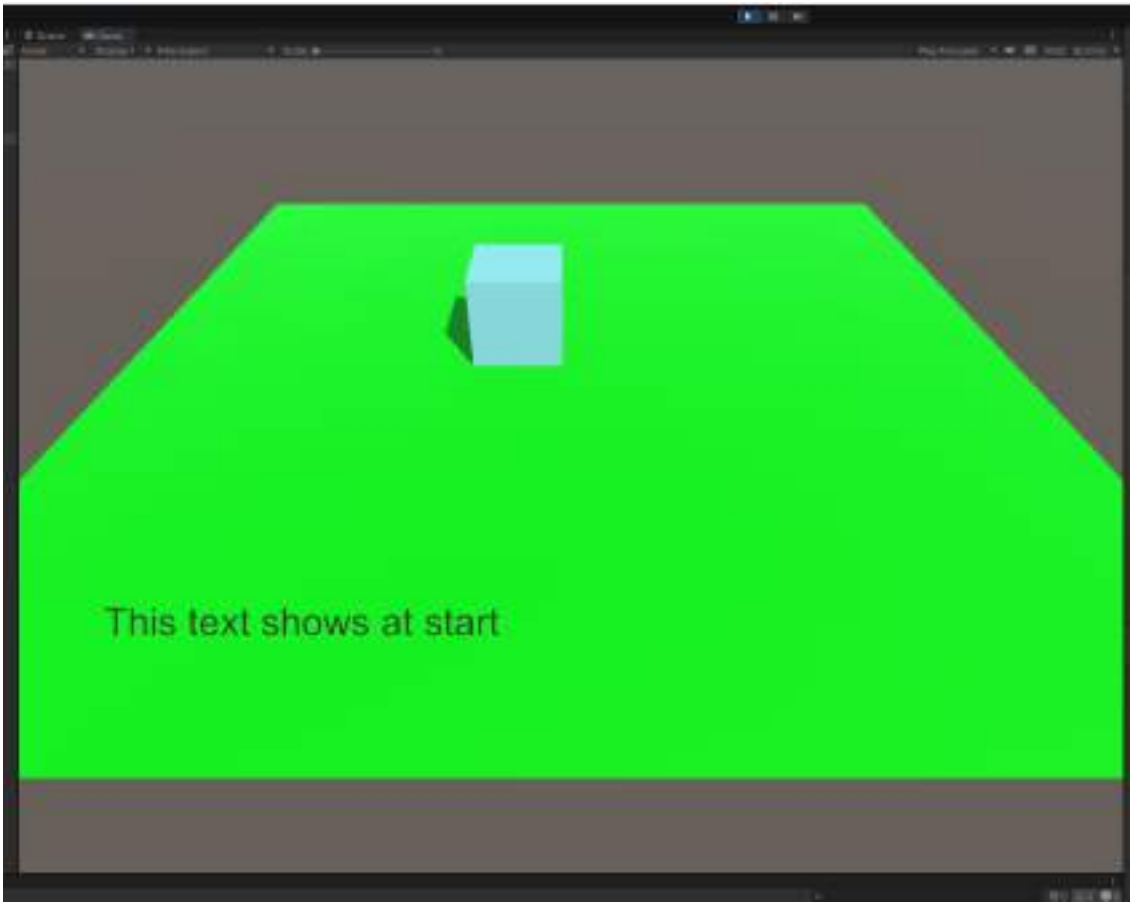
GUARDAR y cerrar el editor de código.

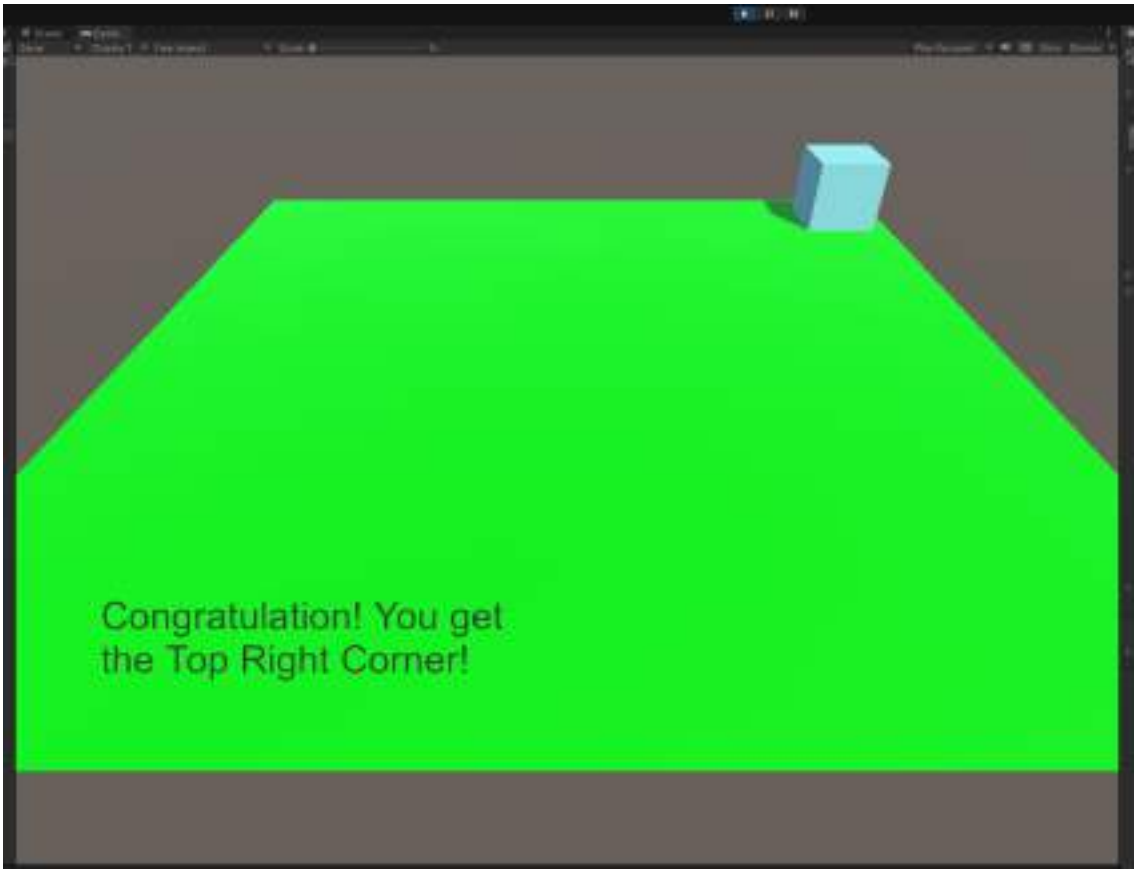
Para cambiar el texto de la interfaz de usuario, debe vincularse al cubo azul claro mediante su panel Inspector.
Establecemos Carácter(script) -> Texto A Cambiar.



Arrastramos Texto (Legado) desde el panel Jerarquía a Carácter(script) -> Texto A Cambiar en el panel inspector del cubo celeste.

GUARDA el proyecto y pulsa play para probar nuestro juego.




Congratulation! You get
the Top Right Corner!

Podemos copiar y pegar el cubo invisible varias veces para crear distintas zonas que muestren textos diferentes.