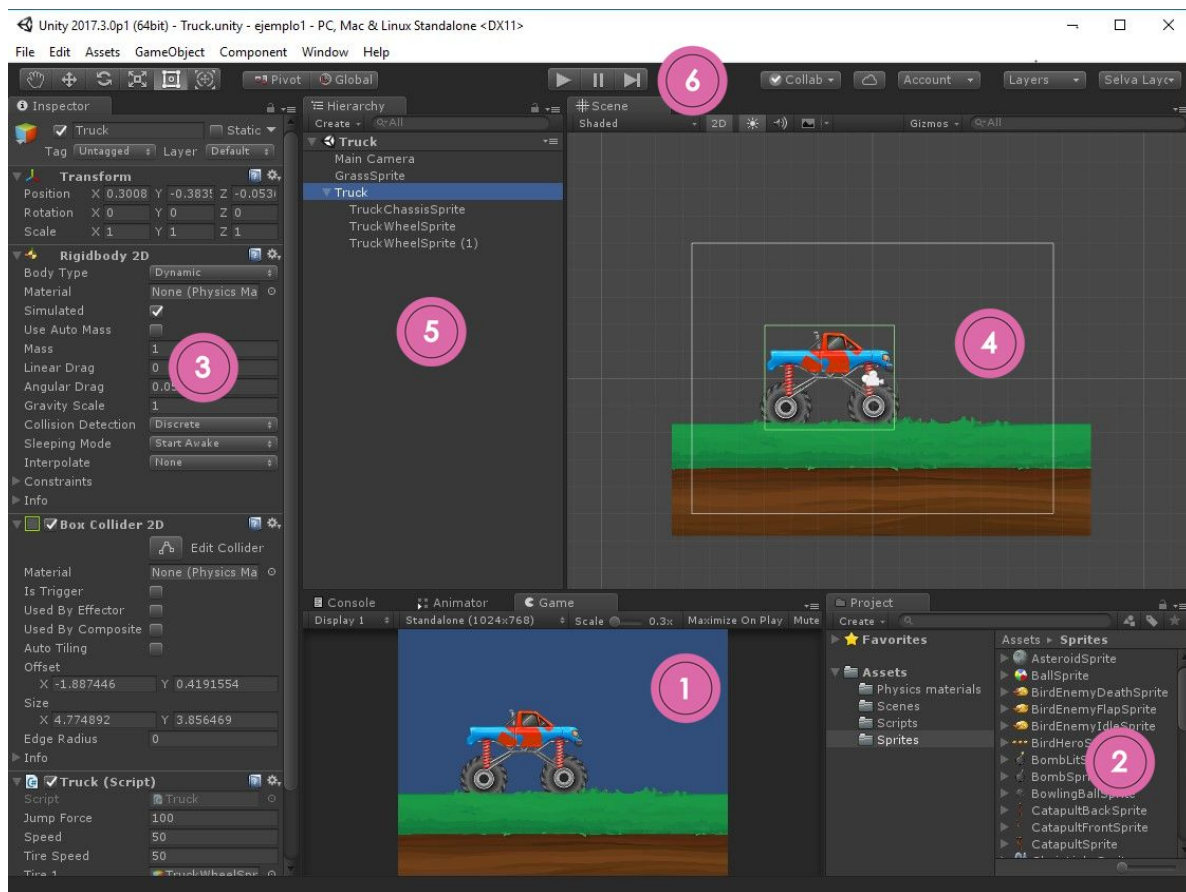


Laboratorio 1: Explorando Unity

Nombre: Jose Daniel Garcia Giron
Carnet: 14152

Parte1: Interfaz de Unity

1. Identifique las partes de la interfaz de Unity asignando uno de los números mostrados en la siguiente imagen a la tabla que se presenta bajo ella.



Número	Descripción
4	Scene view: Nos permite visualizar la escena y colocar los elementos gráficos que deseamos que la formen
6	Toolbar: Tiene la funcionalidad para ejecutar nuestro proyecto. También tiene botones que nos ayudan a navegar en la escena y a mover los objetos dentro de ella
5	Hierarchy window: Nos muestra todos los elementos que forman la escena en una vista

Laboratorio 1: Explorando Unity

	jerárquica. Indica las relaciones de padre-hijos entre los objetos que se encuentran en la escena.
3	Inspector window: Es una ventana que ajusta su contenido dependiendo del objeto que se tenga seleccionado. Lo utilizamos para agregar nuevos componentes a los objetos y modificar los valores de los mismos.
1	Game view: Muestra como se verá nuestro juego
2	Project inspector: Despliega la lista de assets (imágenes, sonidos, scripts de código) que tenemos disponibles para utilizar en nuestro proyecto. El contenido que se muestra corresponde al folder de Assets del proyecto de Unity.

2. Abra El editor de Unity en su computadora y corrobore que todo se haya instalado correctamente. Cree un nuevo proyecto para hacer la prueba.
3. Identifique las partes del editor de Unity en su computadora y re ordene las ventanas si lo desea.
4. Cierre el nuevo proyecto creado

Parte 1: Clonando el repositorio

1. Revisar que se tenga Github desktop instalado. Si no se tiene instalado descargar acá.
<https://desktop.github.com/>
2. Únase al github classroom para esta clase
<https://classroom.github.com/classrooms/38238693-plataformas-2018-seccion-10>
3. Diríjase a la dirección <https://classroom.github.com/a/ixtpHfuO>
4. Acepte la tarea
5. Si aparece esta descripción, esperar a que el repositorio se cree. Cuando esté creado se mostrará una confirmación diciendo Importing complete! También se le enviará un correo con la

Laboratorio 1: Explorando Unity

URL del nuevo repositorio.

Preparing your new repository

There is no need to keep this window open, we'll email you when the import is done.

uvg-cc3055-20/cc3055-20-02-lab-infinite-flappy-alhvi-student

✓ Importing complete! Your new repository [uvg-cc3055-20/cc3055-20-02-lab-infinite-flappy-alhvi-student](#) is ready.

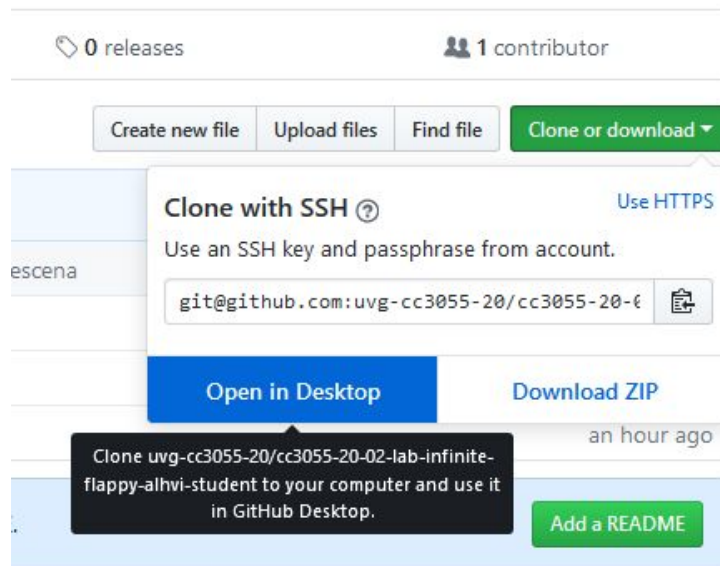
6. Ingresar a la página del repositorio, dando click en el link del mismo. La página del repositorio debe verse como esta

The screenshot shows the GitHub repository page for 'uvg-cc3055-20 / cc3055-20-02-lab-infinite-flappy-alhvi-student'. The repository was created by GitHub Classroom. It has 3 commits, 1 branch, 0 releases, and 1 contributor. The 'master' branch is selected. A table of commits is shown, with the latest commit being 'se elimino elemento innecesario de la escena' by user 'alhvi' 25 minutes ago. The commit history includes 'Assets', 'ProjectSettings', 'UnityPackageManager', and '.gitignore'. A button 'Add a README' is visible at the bottom.

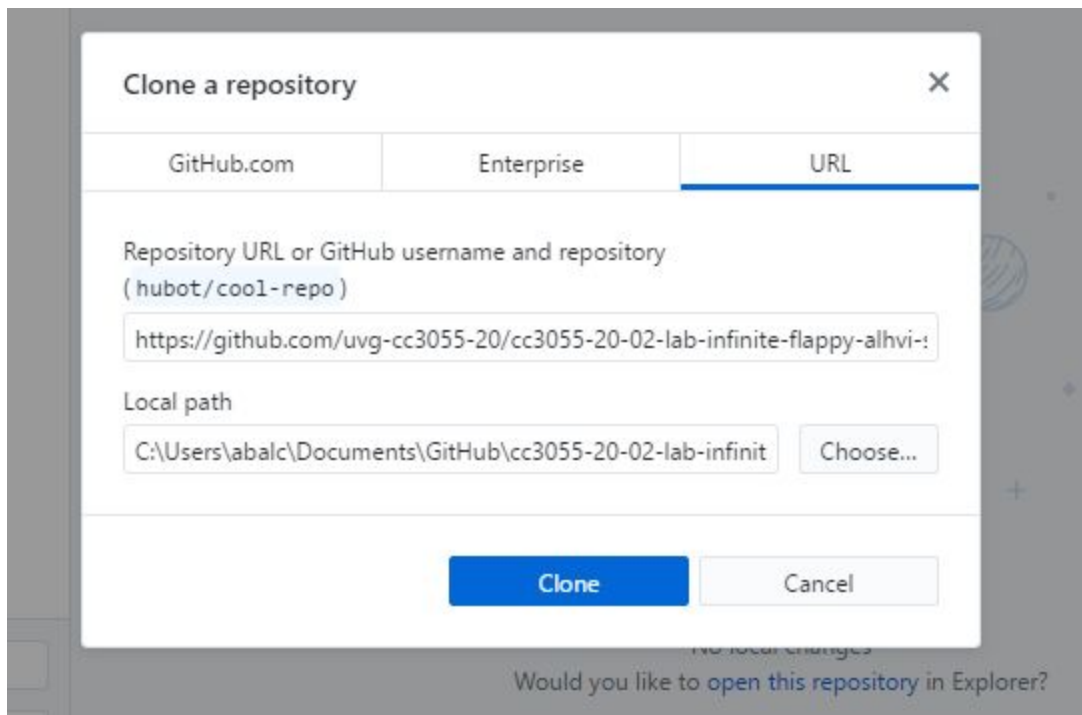
Commit	Message	Time
alhvi	se elimino elemento innecesario de la escena	25 minutes ago
	se agregaron los assets	26 minutes ago
	se agregaron los assets	26 minutes ago
	Initial commit	an hour ago

7. En la página del repositorio, dar click en el botón de clone or download. Después utilizar la opción Open in Desktop, para clonar el repositorio usando Github Desktop.

Laboratorio 1: Explorando Unity



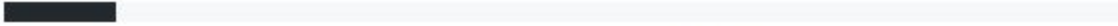
8. En Github desktop debe salir la opción de clonar el repositorio de esta manera.



9. Confirme la ubicación en donde desea clonar el repositorio y de click en el botón de Clone. Debe aparecer el siguiente mensaje.






Laboratorio 1: Explorando Unity

 Cloning cc3055-20-02-lab-infinite-fla...



remote: Compressing objects: 100% (37/37), done.

10. Dirigirse al folder donde se creó el repositorio local y verificar que se hayan descargado los archivos y se haya generado el folder (oculto) .git

is PC > Documents > GitHub > cc3055-20-02-lab-infinite-flappy-alhvi-student				
Name	Date modified	Type	Size	
 .git	1/23/2018 9:48 PM	File folder		
 Assets	1/23/2018 9:46 PM	File folder		
 ProjectSettings	1/23/2018 9:46 PM	File folder		
 UnityPackageManager	1/23/2018 9:46 PM	File folder		
 .gitignore	1/23/2018 9:46 PM	Text Document	1 KB	

11. Abra Unity y seleccione la opción Open y escoja el folder donde se bajó el proyecto.

Si tiene problemas al clonar su repositorio consulte con su profesora.

Parte 2: Desarrollo de un clon de flappy bird sencillo

2.1 Colocando los elementos en la escena

1. Ubique el folder de Assets/Sprites en el Project inspector
2. Arrastre hacia la escena la imagen Bird y colóquela en cualquier posición dentro de la cámara
3. Coloque en la escena la imagen Sky en la posición 24, 0, 0. Si este tapa al objeto Bird, no hay problema, luego lo ordenaremos.
4. Coloque en la escena la imagen Grass en la posición 24, -3, 0

Laboratorio 1: Explorando Unity

2.2 Ordenando los elementos de la escena

1. Seleccione el objeto Sky
2. Teniendo el Sky seleccionado diríjase a el Inspector y al componente Sprite Renderer. En este componente busque la opción Sorting Layer y seleccione Add Sorting Layer.
3. Agregue tres sorting layers en este orden: Background, World y Character. El orden de las sorting layers dice cuál elemento está más atrás.
4. En la Hierarchy view seleccione uno por uno los objetos y agreguelos a las siguientes sorting layers:
 - a. Sky: Background
 - b. Grass: World
 - c. Bird: Character

Revise que los elementos se desplieguen correctamente. Si no es el caso consulte con su auxiliar o profesora.

2.3 Agregando el Rigidbody2D

1. Según lo visto en clase agregue un componente Rigidbody2D al GameObject Bird
2. Ejecute el juego y compruebe que el GameObject Bird responde a la física del motor, cayendo al infinito.

Si no recuerda como realizar este paso o si tiene algún problema consulte con su auxiliar o profesora.

2.4 Agregando Colliders2D

1. Según lo visto en clase agregue un componente BoxCollider2D al objeto Grass y ajuste el tamaño del colisionador. Probablemente sea necesario darle zoom out a la escena para ver correctamente el tamaño.
2. Según lo visto en clase agregue un componente CircleCollider2D al objeto Bird y ajuste el tamaño del colisionador
3. Ejecute el juego y compruebe que el objeto Bird choca contra el suelo y se detiene.

Si no recuerda como realizar este paso o si tiene algún problema consulte con su auxiliar o profesora.

2.5 Agregando un Script

1. Diríjase al Project window y en el directorio Assets cree un folder llamado Scripts. (Click derecho y escoger crear)
2. Dentro del folder llamado Scripts cree un nuevo script de C# llamado BirdScript
3. Seleccione el game object Bird y agréguele el nuevo script como un componente

Laboratorio 1: Explorando Unity

2.6 Editando el Script

1. De doble click sobre el script BirdScript en el Project window para abrir el IDE que nos permita editar el código (Visual Studio o Mono Develop)
2. A modo de prueba escriba dentro de la función Start el código `Debug.Log("Start");`
3. Escriba dentro de la función Update el código `Debug.Log("Update");`
4. Regrese al editor de Unity y corra el proyecto
5. Observe la consola de Unity y reflexione según la forma de correr del script según lo que ve y según los comentarios autogenerados en el script

Si necesita ayuda ubicando la consola o desea comentar sobre la forma de correr del script hable con su auxiliar o profesora.

2.7 Detectando cuando el usuario presiona una tecla

Unity nos proporciona ya un Input manager que detecta cuando el usuario presionó una tecla o un botón del mouse.

Para facilitar el uso del input manager, Unity ya tiene codificadas algunas acciones y ligadas a algunas de las teclas. Por ejemplo la acción "Jump" se encuentra ya ligada a la tecla de barra espaciadora.

1. Elimine el `Debug.Log` que se encuentra en la función de Update
2. Para detectar cuando el usuario presiona la barra espaciadora agregue el siguiente código a la función Update

```
if (Input.GetButtonDown("Jump")) {
    Debug.Log("Espacio");
}
```
3. Regrese a Unity y ejecute el proyecto. Presione la barra espaciadora y observe lo que pasa en la consola.
4. Reflexione sobre la manera en la que se corre el código dentro de la función Update y como el programa detecta que el usuario presiona la tecla.

Si el programa no se comporta de la manera en que debería consulte a su auxiliar o profesora.

2.9 Agregando fuerza hacia arriba al personaje

Para que el personaje pueda volar tendremos que agregarle una fuerza al componente Rigidbody. Haremos esto cuando el usuario presione la barra espaciadora

1. Declare una variable `jumpForce`, como miembro de la clase BirdScript que indique cuánta será la fuerza que aplicaremos al personaje. La variable debe ser de tipo float y ser pública para poder ser accedida en el editor. Le asignaremos a esta variable un valor de 10f.

Laboratorio 1: Explorando Unity

Para poder agregarle la fuerza al rigidbody debemos primero tener una referencia a él en el código

2. Declare una variable que aloje al componente Rigidbody2D. La variable debe ser del tipo Rigidbody2D y declararse como miembro de la clase. Ejemplo:

```
Rigidbody2D rb;
```
3. Elimine el Debug.log de la función Start y agregue el siguiente código para obtener la referencia del Rigidbody2D del objeto Bird

```
rb = GetComponent<Rigidbody2D>();
```
4. Elimine el Debug.Log de la detección de la tecla en la función Update y escriba el código

```
rb.AddForce
```
5. El parámetro de la función será el vector hacia arriba, que puede colocarse como Vector2.up o ya sea new Vector2(0, 1) multiplicado por la variable jumpForce que creamos en el paso 1 de esta sección.
6. Regrese al editor de Unity y compruebe que el personaje se mueve hacia arriba cuando se presiona la tecla barra espaciadora.

Si tiene algún problema al agregar la fuerza al rigidbody consulte a su auxiliar o profesora.

2.10 Mejorando la funcionalidad del flap

Cuando el personaje ya está siendo afectado por la gravedad, la fuerza no es suficiente para elevarlo a donde se esperaba. Por lo que colocaremos la velocidad del Rigidbody2D en 0 antes de aplicar la nueva fuerza.

1. En la función update, como primera instrucción del if (antes del AddForce) coloque

```
rb.velocity = Vector2.zero;
```
2. Diríjase al editor de Unity y compruebe como funciona el salto ahora.

Si el programa no se comporta de la manera en que debería consulte a su auxiliar o profesora.

2.11 Haciendo que el personaje recorra el nivel

Para que el personaje recorra el nivel en X lo moveremos de forma constante. Haremos esto afectando directamente el componente transform del objeto. El componente transform contiene la posición, rotación y escala del objeto.

1. Declare una variable forwardSpeed de tipo float, como miembro de la clase BirdScript. Haremos esta variable pública y le daremos un valor de 2f.
2. En la función Update, afuera del if coloque el código

```
rb.transform.Translate(new Vector3(1, 0, 0) * forwardSpeed * Time.deltaTime);
```
3. Reflexione acerca de los parámetros de la función translate y tome en cuenta que cuando se mueve directamente la posición de un objeto, sin utilizar fuerzas, se debe multiplicar por Time.deltaTime, que es el delta (o diferencia) de tiempo entre cada frame.

Laboratorio 1: Explorando Unity

4. Regrese a Unity y ejecute el programa. Analice lo que pasa en la ventana de Game.

Si tiene alguna duda con respecto a estos pasos consulte a su auxiliar o profesora.

2.12 Haciendo que la cámara siga al personaje

Si todo va bien podemos notar que el personaje se sale del área de la cámara y pasa a no desplegarse en pantalla. Para corregir esto haremos que la cámara siga al personaje. La manera en que haremos esto es afectando directamente el componente transform de la cámara.

Para esto debemos tener acceso al game object de la cámara en nuestro script BirdScript. Haremos esto declarando un objeto dentro del BirdScript de tipo GameObject, que sea público y le asignaremos la cámara.

1. Declare un objeto público, de tipo GameObject, de nombre cam como miembro de la clase BirdScript.
2. En la función Update, aplique la función translate al transform del objeto cam con los mismos parámetros.
3. Intente correr de nuevo el programa.
4. Notará que el código no compila pues necesita una referencia a un objeto. La referencia que necesita es el objeto cam en el componente BirdScript.
5. Seleccione el objeto Bird
6. Busque el componente BirdScript dentro de los componentes de este objeto.
7. Busque en la Hierarchy view el componente Main Camera y arrástrelo a la referencia que necesita BirdScript.
8. Corra de nuevo el proyecto y verifique que funciona correctamente.

Si el proyecto no cumple o no funciona correctamente consulte a su auxiliar o profesora.

2.13 Preparando el nivel

1. Arrastre a la escena el sprite Column y colóquelo en la sorting layer World
2. Agregue a este objeto un BoxCollider2D y ajuste el tamaño del colisionador.
3. Duplique el objeto utilizando el Hierarchy view
4. Dele al nuevo objeto una rotación de 180 en Z utilizando el componente transform en el Inspector
5. Coloque el objeto en un lugar apropiado en la escena.
6. Duplique las columnas utilizando el hierarchy view o los comandos ctrl + c y ctrl + v para crear un nivel.
7. Corra el proyecto y verifique que las columnas tengan su colisionador.

Si tiene alguna duda de como realizar estos pasos consulte a su auxiliar o profesora.

Laboratorio 1: Explorando Unity

2.14 Agregando una condición de derrota

Haremos que el personaje muera cuando tenga una colisión, ya sea contra las columnas o el piso. Para hacer esto utilizaremos la función `OnCollisionEnter2D` de Unity. Ejecutaremos el código en la función `update` solamente si el personaje está vivo.

1. Declare una variable booleana llamada `dead` como miembro de la clase `BirdScript`. Asígnele el valor de `false`.
2. Haga que el contenido de la función `Update` se ejecute solamente si la variable `dead` es falsa.
3. Abajo de la función `update` agregue el siguiente código que es la función que detectará las colisiones.

```
private void OnCollisionEnter2D(Collision2D collision)
{
    // Código para detectar colisiones
}
```

4. Haga que la variable `dead` sea `true` cuando se detecte una colisión.
5. Ejecute el juego y observe que pasa. El personaje y la cámara deberían de dejar de moverse al colisionar con algo.

Si el programa no funciona como debe o tiene alguna duda consulte a su auxiliar o profesora.

2.15 Haciendo que el personaje pare al llegar a cierta posición

Al momento nuestro personaje continúa volando hasta que dejemos de correr el programa. Pero haremos que pare al llegar a cierto punto.

1. Ejecute el programa y pause la ejecución momentos antes de llegar al final.
2. Tome nota de la posición en X del personaje
3. Agregue una condición que diga que el personaje muere al alcanzar esa posición en X
4. Ejecute el juego y pruebe que su código funcione correctamente.

2.16 Guardando la escena

1. En el project inspector cree un folder llamado `Scenes`
2. En el menú de `File`, guarde la escena como `Flappy.unity`
3. En el menú de `File`, guarde el proyecto completo (`Save Project`)
4. Cierre su proyecto, de `commit` y `push` a su repositorio en `Github Classroom`

Entrega: Vía Github Classroom el 14 de abril a las 23:59

Descargue este archivo como PDF y colóquelo en el directorio raíz de su repositorio

Grabe un video con el juego funcionando debe mostrar todas las funcionalidades requeridas

Suba el video a Youtube

Laboratorio 1: Explorando Unity

Haga un archivo Readme.md y colóquelo en el directorio raíz de su repositorio. En este archivo coloque el URL al video.

Darle commit y push al código a su repositorio

Puede omitir el paso del video si termina su laboratorio en clase y lo muestra a su auxiliar o profesora.