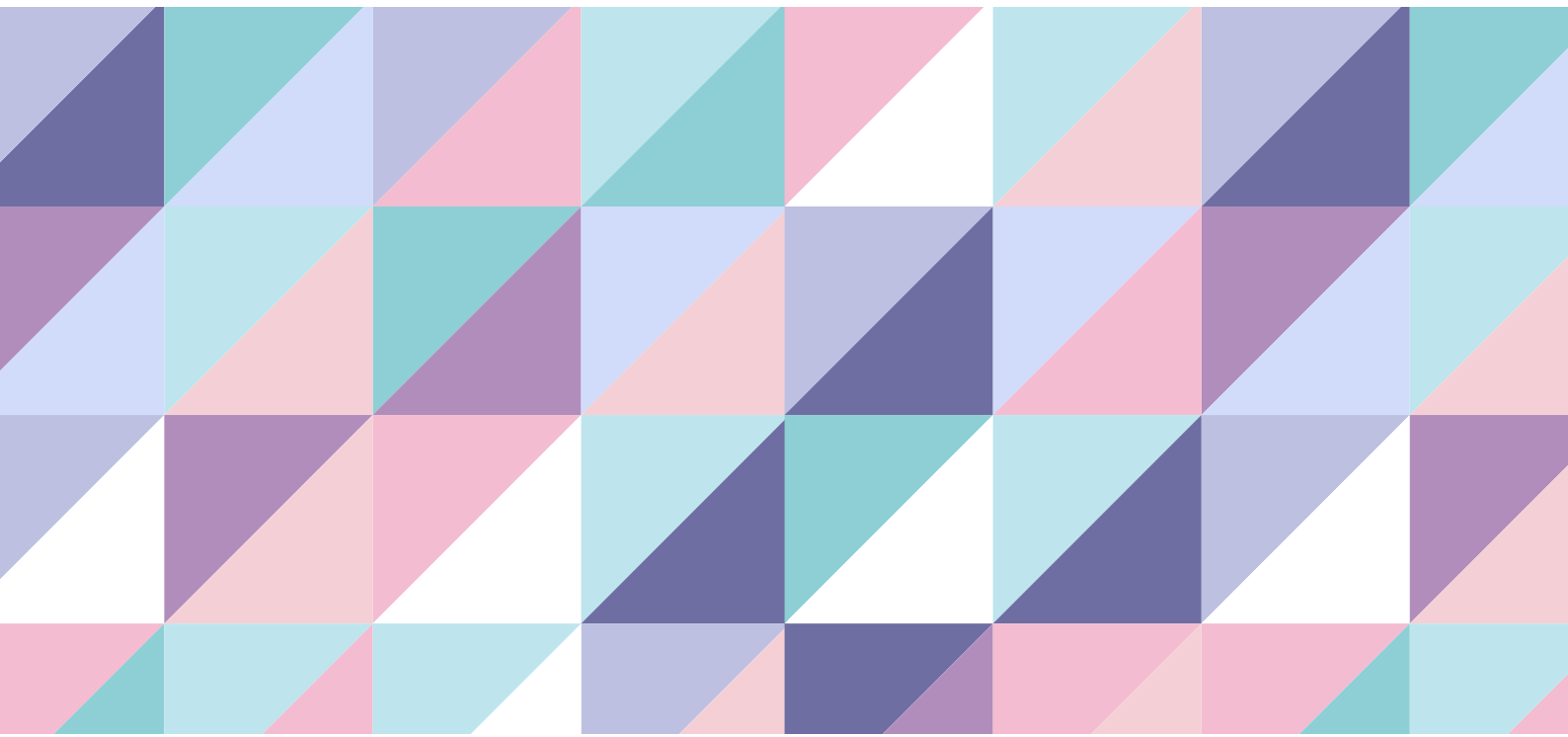


Física para Programadores de Videojuegos 2021-1

Prácticas de Laboratorio

Dra. Eloísa del Carmen García Canseco



Copyright © 2021 E. García–Canseco

PUBLICADO POR FACULTAD DE CIENCIAS, UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

[HTTP://ELOISAGC.NET](http://ELOISAGC.NET)

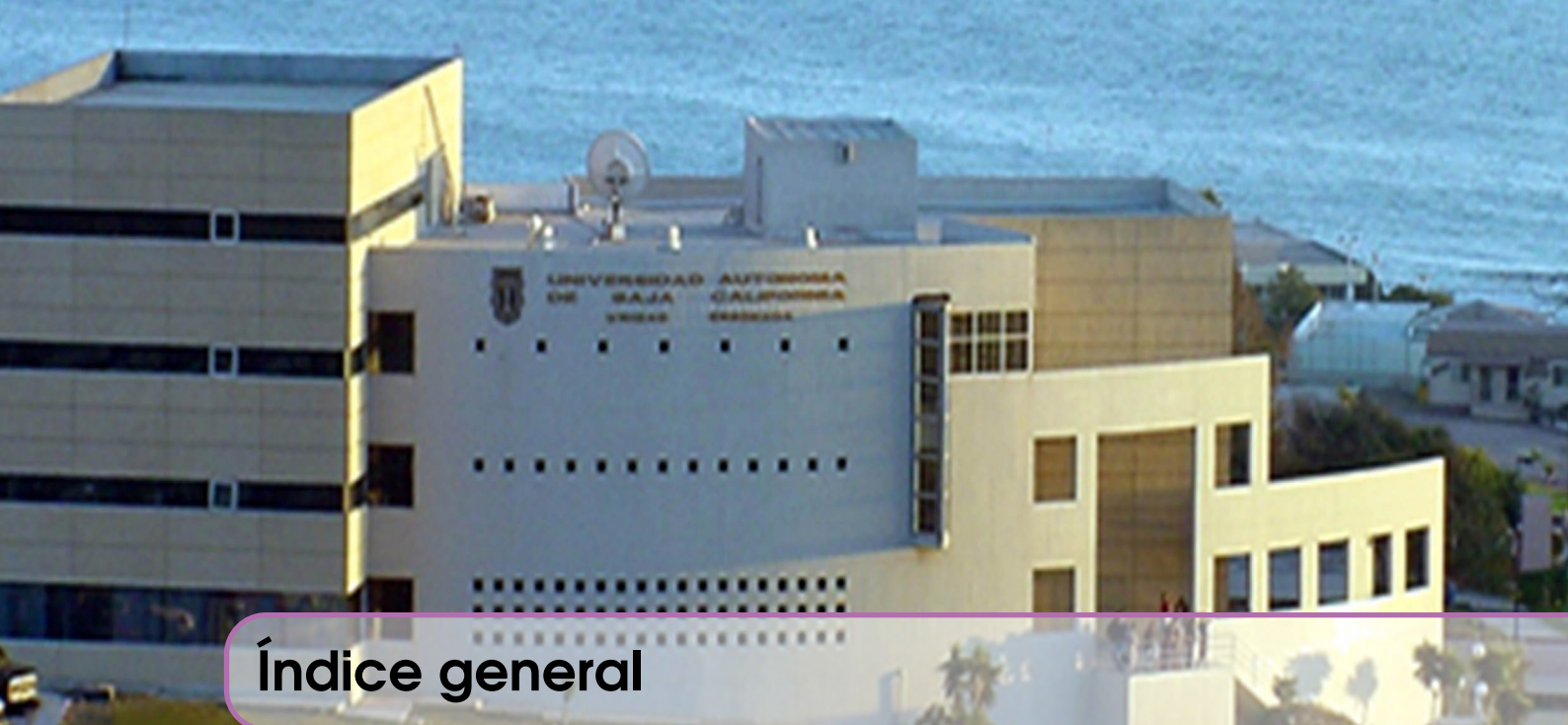
Física para programadores de videojuegos (clave 23863) que se imparte como materia optativa en los planes de estudios 2017-2 de las carreras de licenciado en física y licenciado en ciencias computacionales.

Agradecimientos:

Licensed under the Creative Commons Attribution-NonCommercial 3.0 Unported License (the “License”). You may not use this file except in compliance with the License. You may obtain a copy of the License at <http://creativecommons.org/licenses/by-nc/3.0>. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Hecho con L^AT_EX

Última modificación: 18 de mayo de 2021



Índice general

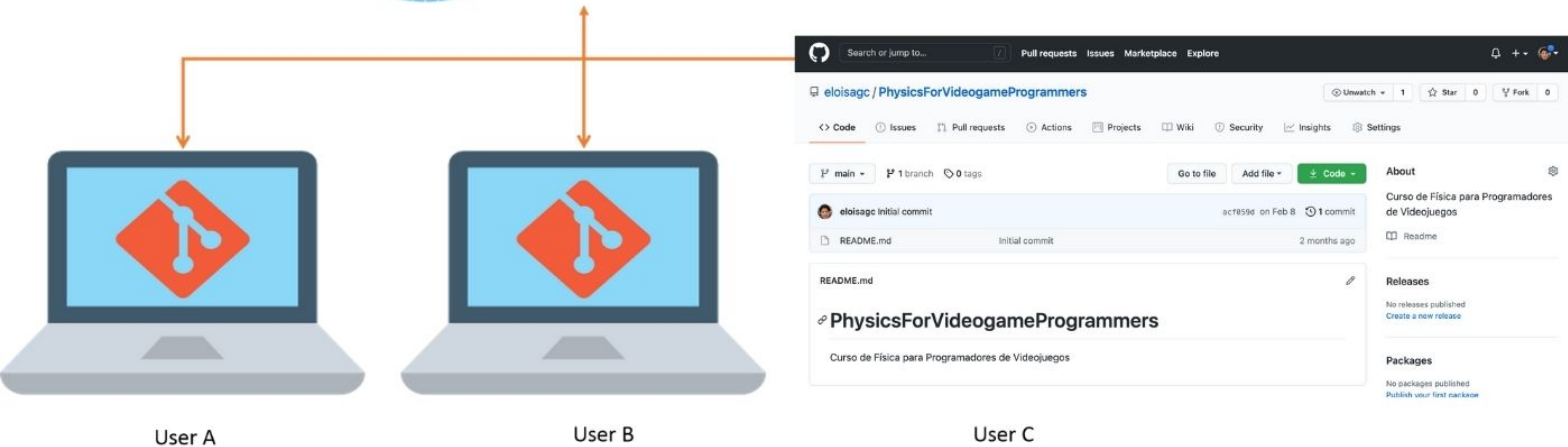
I	Herramientas	
1	Introducción a GitHub	7
1.1	Objetivo general	7
1.2	Fundamentos teóricos	7
1.2.1	Control de versiones	7
1.2.2	Tipos de VCS	7
1.3	Material	7
1.4	Procedimiento	8
1.5	Reporte de resultados	11
II	Cinemática	
2	Movimiento traslacional en 2D	15
2.1	Objetivo general	15
2.2	Fundamentos teóricos	15
2.3	Material	15
2.4	Procedimiento	16
2.5	Reporte de resultados	16
3	Movimiento rotacional	17
3.1	Objetivo general	17
3.2	Fundamentos teóricos	17

3.3	Material	18
3.4	Procedimiento	18
3.5	Reporte de resultados	18
4	Mecánica Newtoniana	19
4.1	Objetivo general	19
4.2	Fundamentos teóricos	19
4.3	Material	19
4.4	Procedimiento	19
4.5	Reporte de resultados	19
	Bibliography	21
	Books	21
	Articles	21
	Páginas web	21



Herramientas

1	Introducción a GitHub	7
1.1	Objetivo general	
1.2	Fundamentos teóricos	
1.3	Material	
1.4	Procedimiento	
1.5	Reporte de resultados	



1. Introducción a GitHub

1.1 Objetivo general

1. Aprender a utilizar el controlador de versiones GitHub

1.2 Fundamentos teóricos

1.2.1 Control de versiones

Un control de versiones [2] (VCS, por sus siglas en inglés de *version control system*) es en esencia una herramienta de software que utilizamos para dar seguimiento a los cambios en el código fuente a lo largo del tiempo. Un VCS permite modificar código en equipo; si se cometen errores, se pueden comparar las versiones y regresar eventualmente a una versión anterior; se protege el código fuente y es multiplataforma. Las principales ventajas de un VCS son:

- Historial de cambios a largo plazo de todos los archivos
- Creación de ramas y fusión de las mismas
- Trazabilidad de cambios

1.2.2 Tipos de VCS

Software	Arquitectura de red	Solución de conflictos	Estatus
Git	Distribuída	Merge	Activo
Mercurial	Distribuida	Merge	Activo
SVN	Ciente-servidor	Merge o lock	Activo
CVS	Cliente-servidor	Merge	Solo mantenimiento

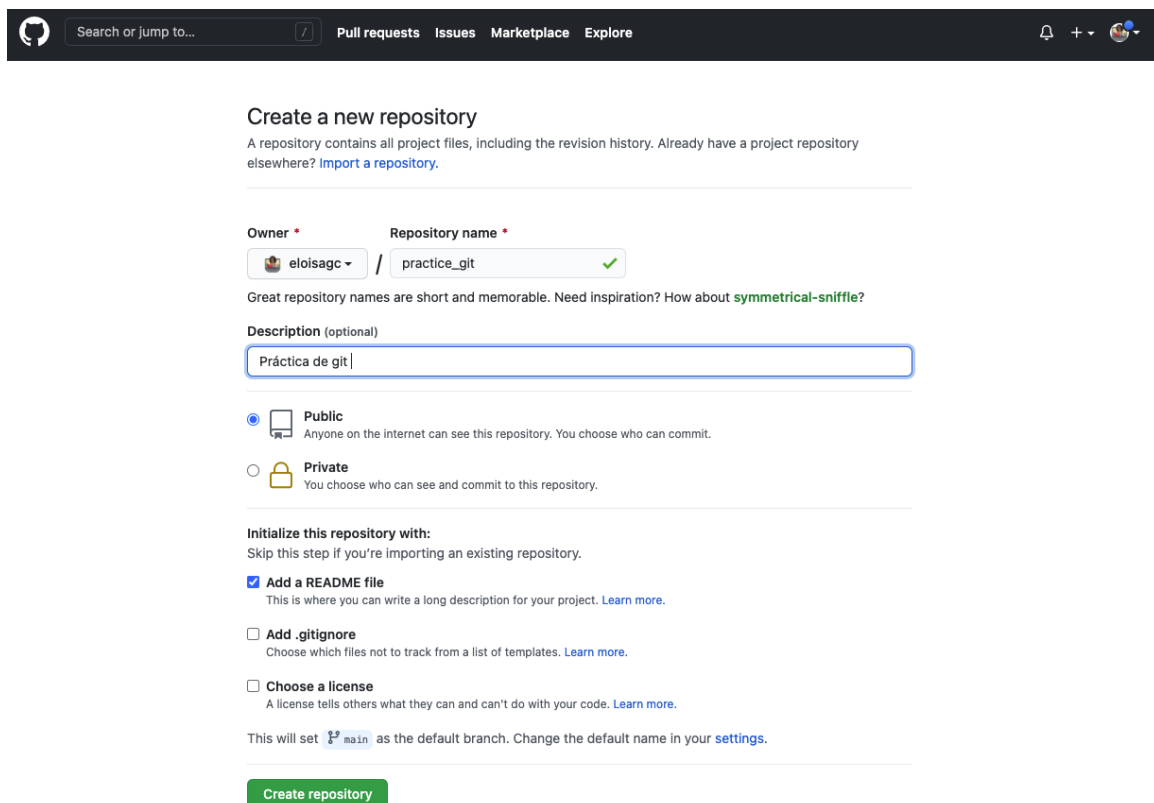
1.3 Material

- Cuenta en Github: <https://github.com/>.
- Terminal o consola para Win/OsX/Linux según corresponda.

- Cliente para Git instalado, por ejemplo: <https://git-scm.com/downloads> o <https://libgit2.org/>.
- *Optional*: Cliente gráfico para Git, por ejemplo: <https://desktop.github.com/>.

1.4 Procedimiento

1. Entra con tu cuenta a la página de Github <https://github.com/>
2. En tu perfil entra a **Your repositories** y en la esquina superior derecha escoge **New** para crear un nuevo directorio
3. Escoge un nombre para tu nuevo directorio, llena la descripción, selecciona tu directorio como **Public** y marca la casilla **Add a README file** como se muestra en la figura 1.1.
4. Una vez que el directorio fue creado, puedes observar que únicamente contiene el archivo **README.mb** (ver figura 1.2), en donde la extensión **.mb** indica un archivo en formato Markdown [1]. Da click en el botón **Code** y copia la dirección web del directorio.
5. Abre la consola o terminal de tu preferencia, ya que a continuación utilizaremos la línea de comandos de git. Con la instrucción **cd** cámbiate al directorio local en el cual trabajarás. Escribe a continuación la instrucción **git clone <https address>** en donde **<https address>** es la dirección de tu directorio en github. Da *enter* para ejecutar la instrucción. Observa lo que ocurre, y al finalizar el proceso ejecuta el commando **ls** para verificar que ya tienes una copia del directorio en tu computadora, como se muestra en la figura 1.3.
6. Ejecuta en la terminal la instrucción **git status**. Esta instrucción la utilizarás cada que quieras revisar cuál es el estado de tu directorio referente a los cambios que se han realizado.
7. Con el editor de tu preferencia crea un nuevo archivo llamado **hello-world.txt**, agrega



Search or jump to...

Pull requests Issues Marketplace Explore

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner **eloisagc** / Repository name **practice_git** ✓

Great repository names are short and memorable. Need inspiration? How about [symmetrical-sniffle?](#)

Description (optional)
Práctica de git

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more](#).

☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more](#).

☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more](#).

This will set **main** as the default branch. Change the default name in your [settings](#).

Create repository

Figura 1.1: Creación de un nuevo directorio en GitHub.

algunas líneas de texto, guarda el archivo y ejecuta nuevamente en la terminal la instrucción `git status`. Observa y analiza el mensaje que se despliega. En la terminal deberías ver algo similar a la pantalla de la figura 1.4

8. Para poder dar seguimiento a los cambios en el archivo `hello-world.txt` hay que agregarlo con la instrucción `git add hello-world.txt`. Ejecuta nuevamente `git status` y observa que se indica ahora que hay un nuevo archivo agregado (figura 1.5).
9. Para salvar los cambios ejecuta ahora la instrucción `git commit -m "archivo hello-world.txt agregado"`, seguida de la instrucción `git status`. ¿Qué ocurre si solo ejecutas la instrucción `git commit` sin el mensaje?
10. Hasta este momento los cambios se han guardado pero solamente en el directorio local en tu computadora. Para sincronizar los cambios con el directorio remoto en Github tienes que ejecutar la instrucción `git push`. Verifica en la página web de Github con tu cuenta personal que se hayan sincronizado los cambios. Debes tener algo similar a lo que se muestra en la figura 1.6
11. Imagina ahora que un miembro de tu equipo creó un nuevo archivo en el directorio de trabajo.

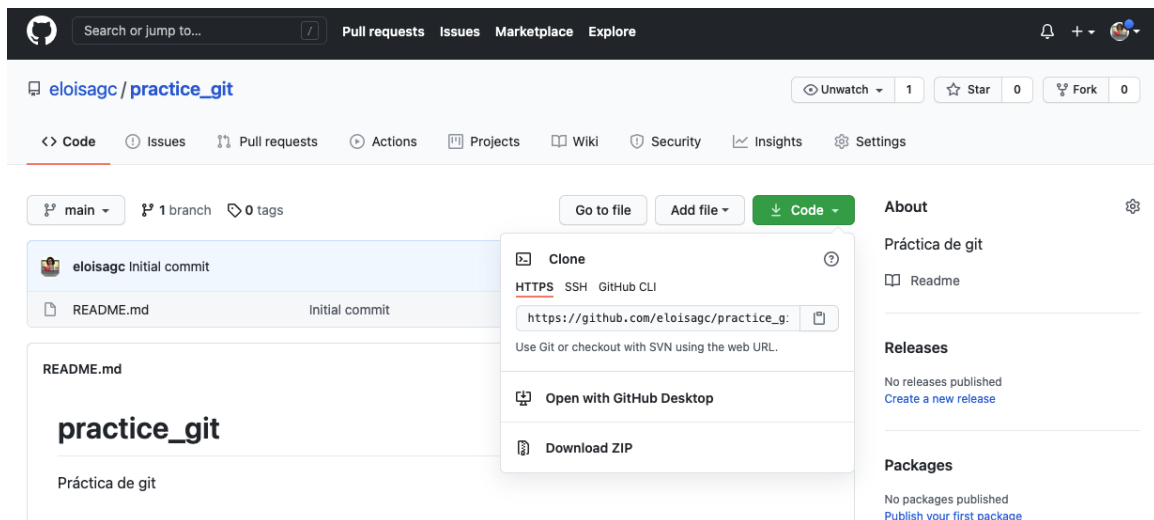


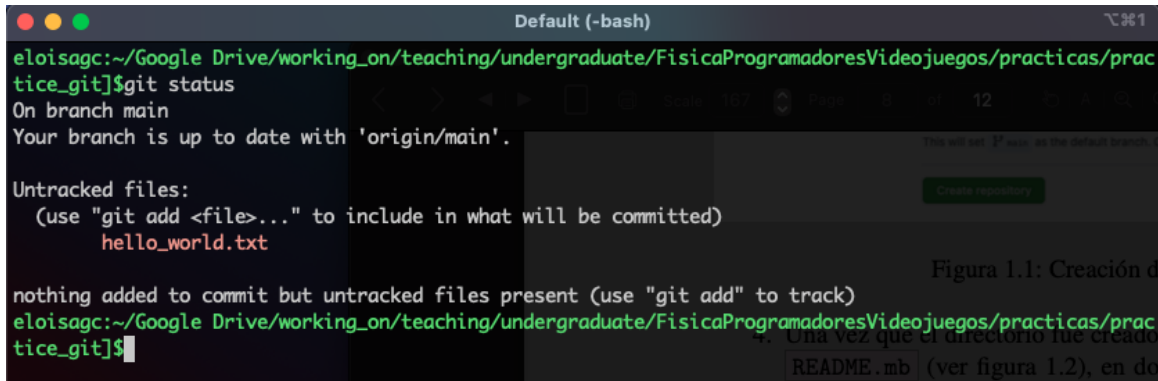
Figura 1.2: Pantalla en GitHub que muestra la dirección https para clonar el directorio recién creado.

```
eloisagc:~/Google Drive/working_on/teaching/undergraduate/FisicaProgramadoresVideojuegos/practicas]$
git clone https://github.com/eloisagc/practice_git.git
Cloning into 'practice_git'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
eloisagc:~/Google Drive/working_on/teaching/undergraduate/FisicaProgramadoresVideojuegos/practicas]$
ls
practice_git
eloisagc:~/Google Drive/working_on/teaching/undergraduate/FisicaProgramadoresVideojuegos/practicas]$
cd practice_git
eloisagc:~/Google Drive/working_on/teaching/undergraduate/FisicaProgramadoresVideojuegos/practicas/p
actice_git]$ ls
README.md
```

Figura 1.3: Terminal en OsX ejecutando la instrucción `git clone` para copiar un directorio remoto en GitHub.

Para simular este hecho, en tu página personal de GitHub, crea un nuevo archivo en el mismo directorio en el que hemos estado trabajando. Puede ser un archivo de texto o código. Guarda los cambios directamente en la página web seleccionando la misma rama principal.

12. Para sincronizar los cambios en tu computadora ejecuta la instrucción `git pull` y posteriormente la instrucción `ls`. Observa que efectivamente has descargado el nuevo archivo.



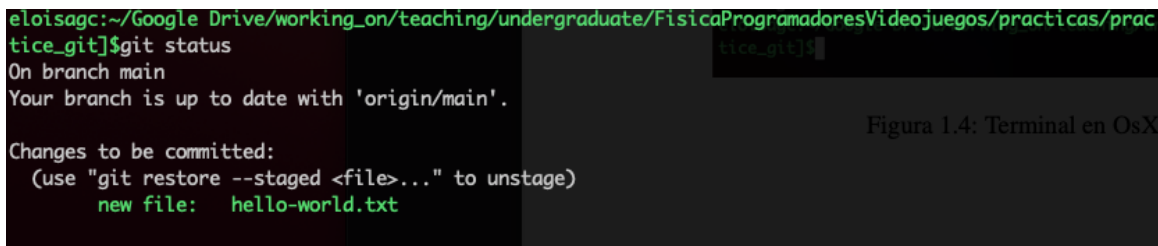
```

eloisagc:~/Google Drive/working_on/teaching/undergraduate/FisicaProgramadoresVideojuegos/practic
tice_git]$git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        hello-world.txt

nothing added to commit but untracked files present (use "git add" to track)
eloisagc:~/Google Drive/working_on/teaching/undergraduate/FisicaProgramadoresVideojuegos/practic
tice_git]$
  
```

Figura 1.4: Terminal en OsX ejecutando la instrucción `git status` en donde se muestra que un archivo no se ha todavía agregado al control de versiones



```

eloisagc:~/Google Drive/working_on/teaching/undergraduate/FisicaProgramadoresVideojuegos/practic
tice_git]$git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   hello-world.txt
  
```

Figura 1.5: Terminal en OsX ejecutando la instrucción `git status` en la que se muestra que se ha agregado un nuevo archivo al control de versiones.

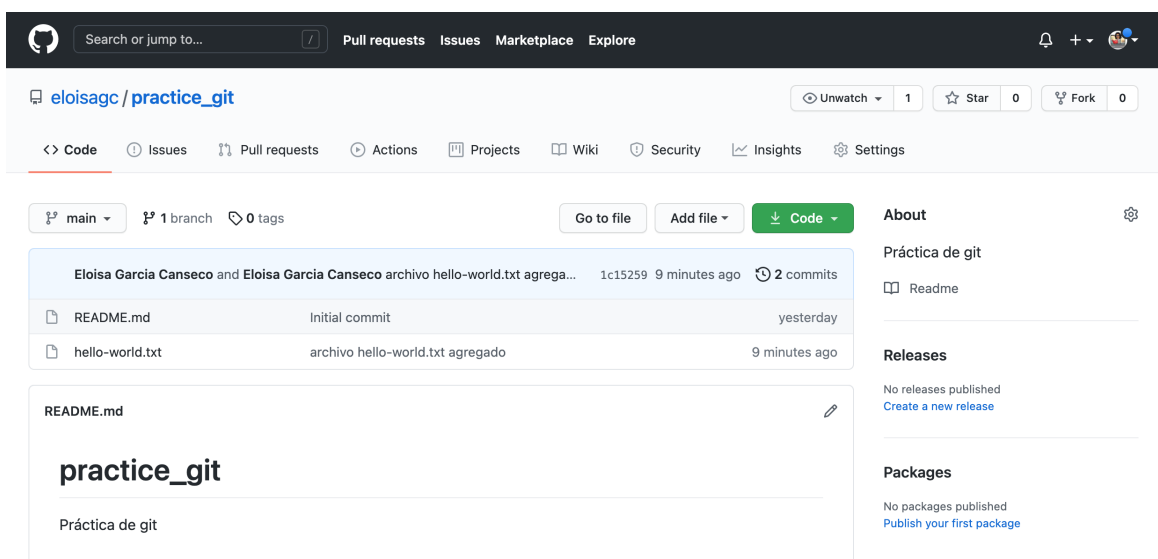


Figura 1.6: Página en GitHub en donde se muestran los cambios ya sincronizados.

13. Por ahora hemos trabajado solo en la rama principal. Puedes ejecutar la instrucción `git branch` y verificar que solo aparece la rama principal, que en este caso se llama `main`. Vamos ahora a crear una nueva rama con la instrucción `git checkout -b feature` de tal forma que sea una rama en la cual explorarás cambios en tus archivos que no quieres que se vean reflejados en la rama principal. Si ejecutas la instrucción `git branch` obtendrás la lista de ramas disponibles en tu directorio, nota que aparece resaltada la rama en la que te encuentras. Observa que puedes volver a la rama principal `main` con la instrucción `git checkout main`.
14. Posiciónate nuevamente en la rama `feature` con la instrucción `git checkout feature`. Edita el archivo `README.md` en tu computadora con el editor de tu preferencia. Guarda los cambios y ejecuta la instrucción `git status`, observarás que hay cambios pendientes por sincronizar. Ejecuta sucesivamente `git add README.md` y `git commit -m 'archivo README.md actualizado'`. Es importante resaltar que los cambios en el archivo `README.md` solo están guardados en la rama `feature` y no en la rama principal. Para verificar esto cámbiate a la rama principal con la instrucción `git checkout main`, abre el archivo `README.md` y observa que corresponde al archivo original.
15. Para sincronizar los cambios en el directorio remoto ejecuta `git push --set-upstream origin feature`. En tu página de GitHub observa que aparece ya la rama creada y que puedes incluso comparar los cambios. Compara los cambios que se muestran en la página web con los que se muestran en la terminal al ejecutar la instrucción `git diff feature`.
16. En tu página de Github, da click en el bottom que dice `Compare&Pull request`, escribe algunos comentarios sobre las modificaciones que hiciste al archivo `README.md` y realiza un pull-request. Observa los mensajes que se despliegan. En la misma página web puedes combinar las ramas, es decir hacer un `merge`. Observa ahora que los cambios realizados en el archivo `README.md` aparecen ya en la rama principal. También hubiéramos podido combinar los cambios directamente en la terminal. Para esto había que posicionarse primero en la rama principal, y posteriormente ejecutar la instrucción `git merge feature`.
17. Sincroniza tu directorio local con la instrucción `git pull`.

1.5 Reporte de resultados

En tu directorio de la práctica en tu página de GitHub, en el menú Wiki crea una página y describe detalladamente todo tu procedimiento. Agrega las capturas de pantallas que consideres conveniente.



Cinemática

2	Movimiento traslacional en 2D	15
2.1	Objetivo general	
2.2	Fundamentos teóricos	
2.3	Material	
2.4	Procedimiento	
2.5	Reporte de resultados	
3	Movimiento rotacional	17
3.1	Objetivo general	
3.2	Fundamentos teóricos	
3.3	Material	
3.4	Procedimiento	
3.5	Reporte de resultados	
4	Mecánica Newtoniana	19
4.1	Objetivo general	
4.2	Fundamentos teóricos	
4.3	Material	
4.4	Procedimiento	
4.5	Reporte de resultados	
	Bibliography	21
	Books	
	Articles	
	Páginas web	



2. Movimiento traslacional en 2D

2.1 Objetivo general

1. Implementar en una plataforma de desarrollo de videojuegos (*game engine*) algunos conceptos de la cinemática, en particular, el movimiento traslacional en 2D.

2.2 Fundamentos teóricos

Considera el movimiento de una partícula en 2 dimensiones. Sea el vector desplazamiento \mathbf{p} definido como $\mathbf{p} = [p_x \ p_y]^T$, donde p_x y p_y son las componentes escalares del vector \mathbf{p} en las direcciones de los ejes x y y respectivamente. Sea el vector velocidad \mathbf{v} definido como $\mathbf{v} = [v_x \ v_y]^T$, donde v_x y v_y son las componentes escalares del vector \mathbf{v} en las direcciones de los ejes x y y respectivamente. Sea el vector aceleración \mathbf{a} definido como $\mathbf{a} = [a_x \ a_y]^T$, donde a_x y a_y son las componentes escalares del vector \mathbf{a} en las direcciones de los ejes x y y respectivamente. Además

$$\mathbf{v} = \frac{d\mathbf{p}}{dt}, \quad \mathbf{a} = \frac{d\mathbf{v}}{dt}$$

Considera que estás diseñando un videojuego en 2D en donde la tarea es disparar un rifle de postas en una feria (Figura 2.1). Necesitas encontrar cuál es la caída vertical de la posta a partir del punto objetivo hasta el punto de contacto. Por ahora vamos a asumir que no hay viento ni resistencia del aire (esto lo veremos más adelante cuando estudiemos el movimiento de proyectiles) por lo que el problema se reduce a un problema de movimiento con aceleración constante, la cual en este caso se debe a la gravedad.

En el momento del disparo, el proyectil sale con una velocidad inicial $\mathbf{v}_0 = [v_0 \ 0]^T$ y aceleración inicial $\mathbf{a}_0 = [0 \ -g]^T$ donde $g = 9,81 \text{ m/s}^2$

2.3 Material

- Cuenta en la página de Github <https://github.com/>
- Plataforma de desarrollo de videojuegos instalada en tu computadora

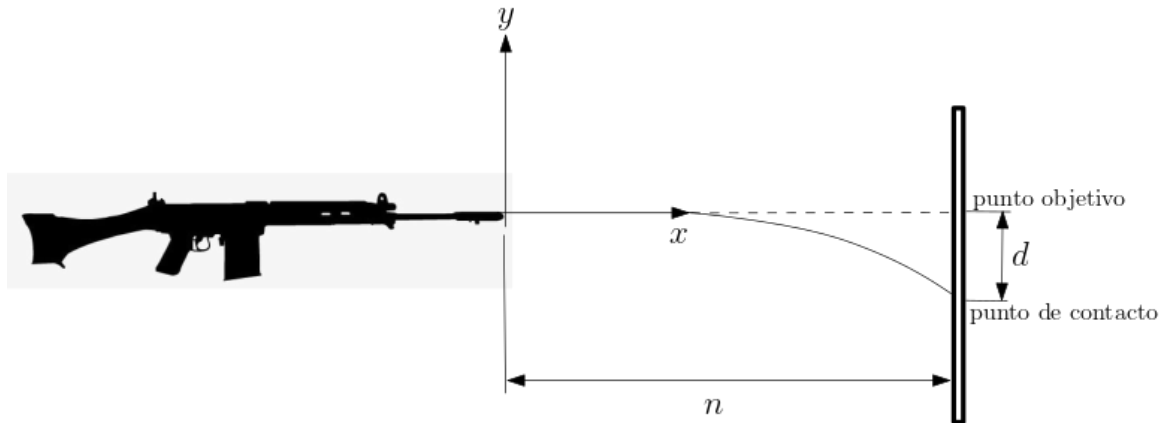


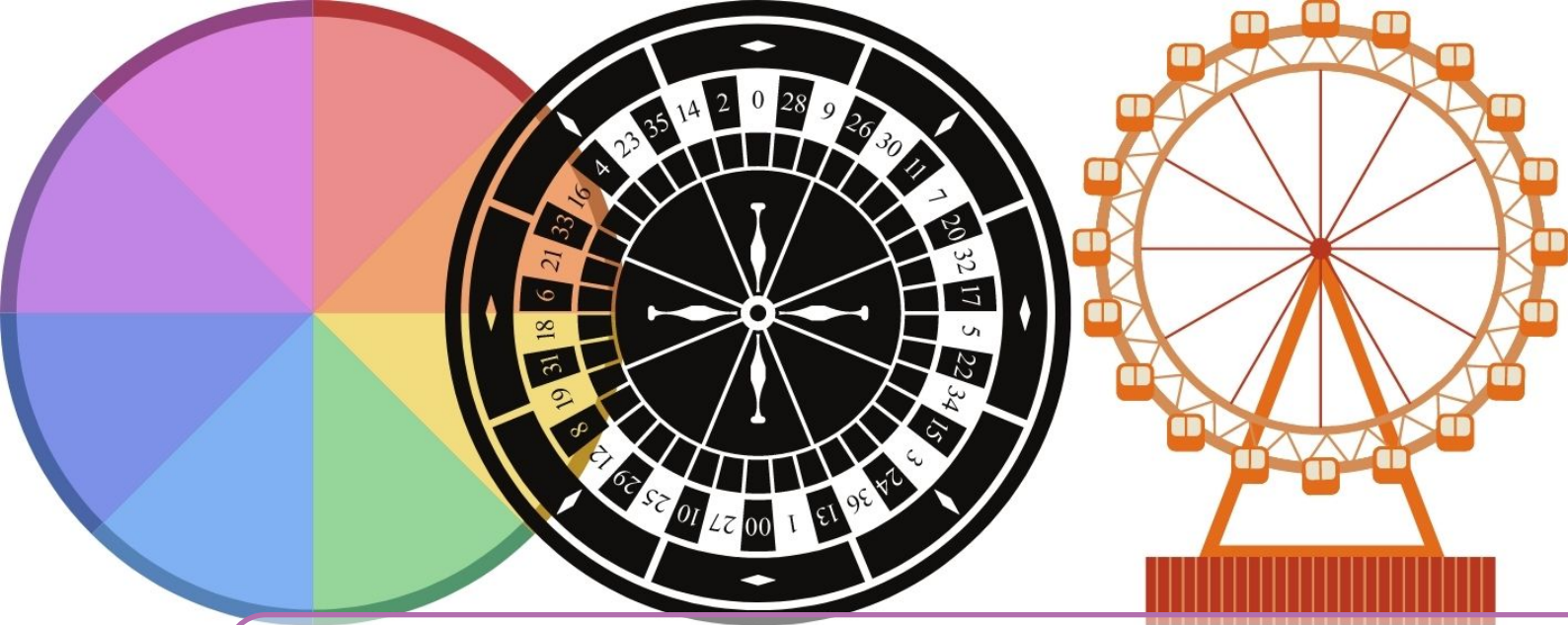
Figura 2.1: La trayectoria de la posta de un rifle puede reducirse a un problema de cinemática en dos dimensiones.

2.4 Procedimiento

1. Encuentra las ecuaciones de la posición $\mathbf{p}(t)$, velocidad $\mathbf{v}(t)$ y aceleración $\mathbf{a}(t)$ instantáneas para el problema de la figura 2.1.
2. Encuentra la caída vertical de la posta d
3. En la plataforma de diseño de videojuegos de tu elección, simula un videojuego en 2D en donde la tarea sea disparar un rifle de postas en una feria en la cual el usuario indique la distancia a la que se encuentra el objetivo y pueda escoger una velocidad inicial.

2.5 Reporte de resultados

Haz un directorio de la práctica en tu página de GitHub, en el menú Wiki crea una página y describe detalladamente todo tu procedimiento. Agrega las capturas de pantallas que consideres conveniente así como el código en donde se vean las ecuaciones de movimiento que implementaste. Incluye un video con el resultado de tu implementación.



3. Movimiento rotacional

3.1 Objetivo general

1. Implementar en una plataforma de desarrollo de videojuegos (*game engine*) algunas de las ecuaciones de la cinemática del movimiento rotacional en 2D.

3.2 Fundamentos teóricos

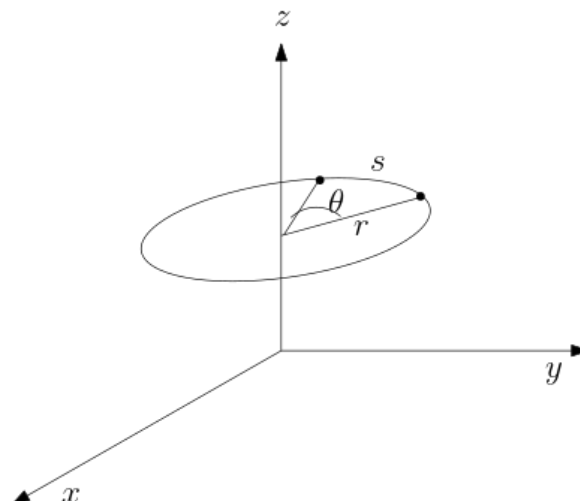


Figura 3.1: Movimiento rotacional de una partícula con respecto a un eje.

Ecuaciones importantes:

- Posición angular: $\theta = \frac{s}{r}$ [rad] con s la longitud de arco medida a partir de la posición angular.
- Desplazamiento angular: $\Delta\theta = \theta_2 - \theta_1$
- Velocidad angular promedio: $\omega_{avg} = \frac{\Delta\theta}{\Delta t}$
- Velocidad angular instantánea: $\omega(t) = \lim_{\Delta t \rightarrow 0} \frac{\Delta\theta}{\Delta t} = \frac{d\theta}{dt}$

- Aceleración angular promedio: $\alpha_{avg} = \frac{\Delta\omega}{\Delta t}$
- Aceleración angular instantánea $\alpha(t) = \lim_{\Delta t \rightarrow 0} \frac{\Delta\omega}{\Delta t} = \frac{d\omega}{dt}$

3.3 Material

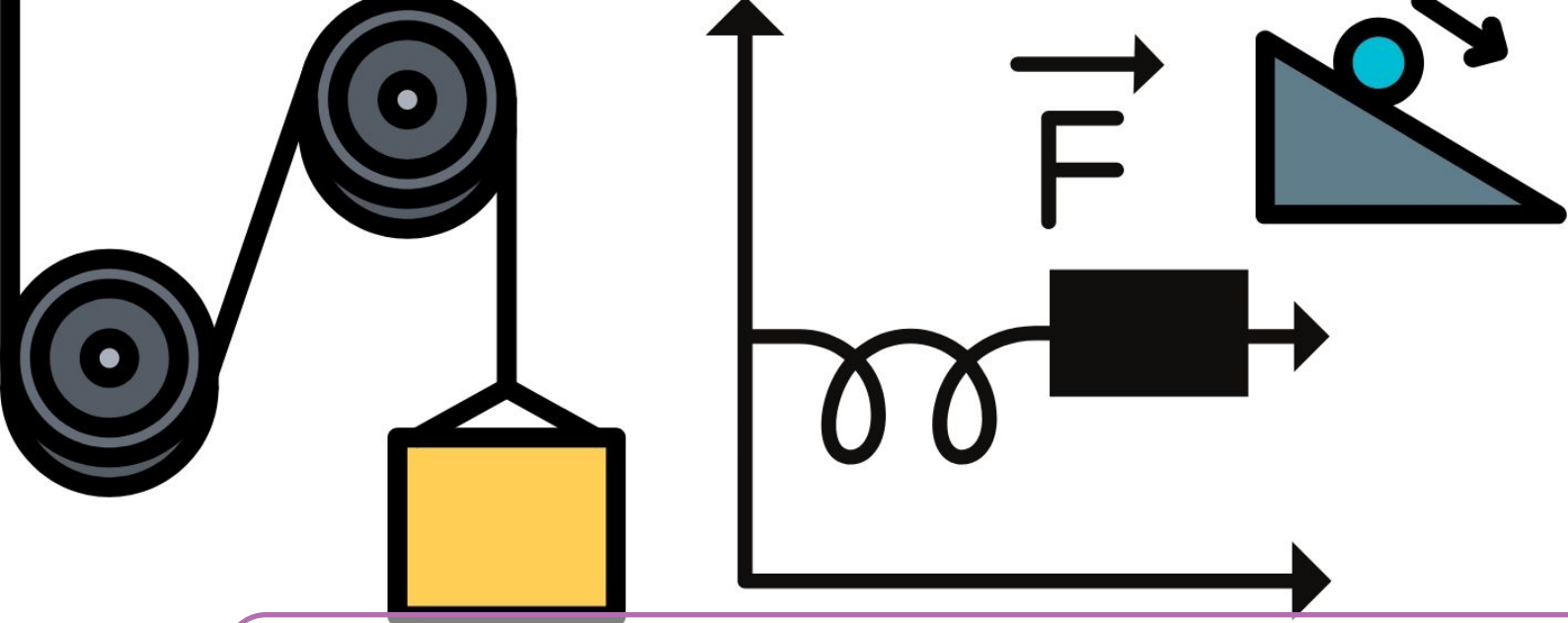
- Cuenta en la página de Github <https://github.com/>
- Plataforma de desarrollo de videojuegos instalada en tu computadora

3.4 Procedimiento

1. Considera que estás diseñando un videojuego en 2D en donde la tarea es ilustrar el movimiento rotacional de la rueda de la fortuna. En este ejercicio no consideraremos las fuerzas que provocan dicho movimiento. Supon que la posición angular de tu asiento está dada por $\theta(t)$ y que la rueda se mueve de acuerdo a la ecuación $\theta(t) = -1 - 0,6t + 0,25t^2$. Simula dicho movimiento en la plataforma de diseño de videojuegos de tu elección.

3.5 Reporte de resultados

Haz un directorio de la práctica en tu página de GitHub, en el menú Wiki crea una página y describe detalladamente todo tu procedimiento. Agrega las capturas de pantallas que consideres conveniente así como el código en donde se vean las ecuaciones de movimiento que implementaste. Incluye un video con el resultado de tu implementación.



4. Mecánica Newtoniana

4.1 Objetivo general

1. Implementar en una plataforma de desarrollo de videojuegos (*game engine*) situaciones que involucren las Leyes de Newton.

4.2 Fundamentos teóricos

4.3 Material

- Cuenta en la página de Github <https://github.com/>
- Plataforma de desarrollo de videojuegos instalada en tu computadora

4.4 Procedimiento

1. Considera que estás diseñando un videojuego en 2D en donde la tarea es ilustrar el desplazamiento de un bloque sobre un plano inclinado (ver figura 4.1)¹. En el videojuego se debe poder cambiar la inclinación del plano inclinado, así como la masa del bloque y el material del plano (es decir, explorarás diferentes tipos de coeficientes de fricción)

4.5 Reporte de resultados

Haz un directorio de la práctica en tu página de GitHub, en el menú Wiki crea una página y describe detalladamente todo tu procedimiento. Agrega las capturas de pantallas que consideres conveniente así como el código en donde se vean las ecuaciones de movimiento que implementaste. Incluye un video con el resultado de tu implementación.

¹Figura tomada de: https://static.wikia.nocookie.net/hdps/images/6/6f/Inclined_Plane_and_Simple.jpg/revision/



Figura 4.1: Desplazamiento de un bloque en un plano inclinado



Bibliografía

Books

Articles

Páginas web

- [1] *Markdown Guide*. URL: <https://www.markdownguide.org/getting-started/> (véase página 8).
- [2] *Qué es el control de versiones*. URL: <https://www.atlassian.com/es/git/tutorials/what-is-version-control> (véase página 7).

