ESTÁNDAR DE COMMIT

MIGUEL ÁNGEL CASTILLO SÁNCHEZ

30/08/2016

¿Que es Git?

Git es una herramienta de software que ayuda a realizar un control de versiones de archivos, de manera distribuida[1],

¿Qué es un repositorio?

Un repositorio en **git** es una "carpeta" donde se guardan todos los archivos relacionados con él y registra un historial de cambios lo que nos permite trasladarnos a un punto específico para retomar algo que se hizo anteriormente o simplemente para revisarlo[2].

Empezar a colaborar

Para empezar a colaborar en el proyecto lo primero a realizar es **clonar** el repositorio en el cual se va a contribuir, usando el comando **git clone url,** dónde **url** es la liga de GitHub donde se encuentra alojado el repositorio.

Por ejemplo:

Si clonamos el repositorio de *naktan-modelado* git descargará una copia de todo el repositorio en tu computadora.

```
git lfs clone https://github.com/BalamChaac/Naktan
```

¿Que es un Commit?

Un commit es una aportación al repositorio en el que estás colaborando. Git realiza una huella digital de la aportación[3] ; registra el autor, fecha y mensaje de commit.

```
commit 71751065138d11da52946586bbb26c0102b214b7
Merge: 0a788bf 0bf422b
Author: Miguel Ángel <BalamChaac@users.noreply.github.com>
```

[1] Cada persona tiene una copia local del repositorio.
[2] Revisar mas acerca de git y su funcionamiento en: https://git-scm.com/about

[3] Git utiliza el algoritmo SHA-1

```
Date:   Wed Aug 24 19:28:24 2016 -0500

    Merge pull request #42 from wf192/master
    Mecánicas de salto y correr implementadas
```

Porqué documentar commits

Los commits ayudan saber qué cambios se realzaron en determinada fecha y porque se hicieron. También ayudan a mejorar el proceso de revisión, a escribir una buena nota de versión y a las nuevas personas que se integran al equipo incluso a nosotros mismos a saber el porqué de nuestras aportaciones.

Malas prácticas de comentarios:

```
git log --oneline  -20

2e342f3 fix commit
d803651 Merge pull request #39 from wf192/master
b0d62fc Akbal fue portado a una maquina de estados mas solida
d99bab2 Akbal implementa una maquina de estados menos casada con
la implementacion fisica
bd99026 Small progress
```

Buenas prácticas de comentarios:

```
git log --oneline  -20

2e342f3 refactor: Dejar el estado de HEAD
d803651 Merge pull request #39 from wf192/master
b0d62fc feat: Portar Akbal a nueva máquina de estados
d99bab2 feat: Separación modular de Akbal es menos dependiente
del motor físico
bd99026 feat: Agregar bandera JUMPING a Character
```

De esta manera revisar lo que pasó hace meses y el porqué de cada uno de los cambios, las mejoras o el arreglo de errores se vuelve posible y eficiente.

Al ser un proyecto largo en algún momento se requerirá de mantenimiento, y tener documentado los commits hará más fácil esta tarea.

Reglas para tener increíbles mensajes de COMMIT

#1 Coloca una etiqueta

Las etiquetas en el título del commit sirven para introducir una idea rápida acerca del commit.

- **feat:** Cuando se trate de una nueva característica, se puede sustituir por el nombre en lo que se esté trabajando como character, player, akbal.
- **fix:** Cuando se solucionó un error, se deberá de hacer referencia al número de error en el pie del mensaje, si es necesario hacer referencia a otra tarea o a un commit en especial.
- **docs:** Cuando se agregue documentación o se realzaron cambios.
- **style:** En el caso de **código** cuando se agregan comas, puntos, se alinea al estilo de codificación. El caso de las demás áreas será cuando se adapten los archivos al estilo especificado en cada estándar.
- **refactor:** Cuando se hace una refactorización del código de producción, en caso de que se haga algún cambio de nombres o alguna

reestructuración interna como optimización, sin cambiar su comportamiento externo.
- **test:** Cuando se agregan pruebas como: unit test, black box, white box, etc.
- **chore:** Actualización del sistema de construcción de código.
- **enhance:** Cuando se hace una mejora a lo ya creado con anterioridad.

Las etiquetas deberán de escribirse en minúsculas seguido de dos puntos y un espacio.

En el caso de artistas podrán usar estas etiquetas, según la categoría a la que pertenezca el trabajo que están realizando:

- **storyboard**
- **standard**
- **concept-art**
- **modeling**
- **rigging**
- **typography**
- **editorial**

Ejemplo:

```
b0d62fc feat: Portar Akbal a nueva máquina de estados
```

#2 Limita el número de caracteres a 50 para el título

Escribir en un máximo de 50 caracteres el título del commit, esto asegura que el mensaje sea leíble y fuerza al colaborador a pensar en una forma muy concisa acerca de lo que realizó en el commit, por esta razón el título del commit deberá ser una síntesis de la aportación.

Ejemplo:

```
bd99026 feat: Add flag JUMPING into Character
```

#3 Separa el título del cuerpo por una línea en blanco

Los mensajes de commit deberán de llevar cuerpo cuando sea necesario explicar el contexto, ¿qué o por qué?

Ejemplo:

```
commit 0bf422b33fd0263fe93cb9085ab7e8457350bfe3
Author: wf192 <walberth.91@gmail.com>
Date:   Wed Aug 24 13:24:32 2016 -0500

    Set Akbal with the storyboard #2

    Akbal meets the specification stablished on storyboard #2,
it has a custum state machine because this character has an
unique behavior.
```

#4 Utiliza lenguaje imperativo

Escribir con un lenguaje imperativo es como dar una instrucción u orden, sólo para el título del commit.

Ejemplo:

```
Limpia tu cuarto
Actualiza tareas en taiga.io
```

Puede parecer un poco agresivo al inicio pero se aplica para seguir las convenciones hechas por git. Por otro lado esta forma de escribir los commits ayuda a evitar que en el cuerpo del mensaje se escribe como una descripción.

Los títulos de commit debe de completar la oración:

If applied, this commit will release version 1.0

#5 Mensajes en inglés sólo para el título el cuerpo en español

Los mensajes hechos por investigadores y artistas el título deberá ser escritos en inglés y el cuerpo en español.

Los programadores deberán de escribir todo el mensaje en inglés.

#6 Limita los caracteres por línea a 72 caracteres

El cuerpo del mensaje de commit no deberá de pasar los 72 caracteres por línea en el commit, para esto se podrán ayudar de un editor de textos[4] configurado a esta regla para facilitar el trabajo.

#7 Usa el cuerpo del mensaje para el ¿qué?, ¿porqué? en lugar del ¿cómo?

Un gran ejemplo es este commit de Bit Coin

```
commit eb0b56b19017ab5c16c745e6da39c53126924ed6
Author: Pieter Wuille <pieter.wuille@gmail.com>
Date:   Fri Aug 1 22:57:55 2014 +0200

Simplify serialize.h's exception handling

Remove the 'state' and 'exceptmask' from serialize.h's stream
implementations, as well as related methods.

As exceptmask always included 'failbit', and setstate was always
called with bits = failbit, all it did was immediately raise an
exception. Get rid of those variables, and replace the setstate
with direct exception throwing (which also removes some dead
code).

As a result, good() is never reached after a failure (there are
only 2 calls, one of which is in tests), and can just be
replaced by !eof().

fail(), clear(n) and exceptions() are just never called. Delete
   them..
```

El cuerpo del mensaje debe de responder ¿qué se cambió? Y ¿porqué se cambió?, el ¿cómo se hizo? va dentro de los comentarios del código.

En el caso de los artistas escribir algunos detalles de cómo realizaron cierta acción o ya sea algún tipo de comando, técnica, o algo que hayan descubierto. Si es algo que debería ir en los estándares de arte agregarlos en estándar correspondiente.

#8 Pie de commit (opcional)

Se agrega un pie de mensaje cuando sea necesario hacer referencia a otro commit, a algún colaborador, a una tarea o error.

```
Arregla el error: #33
Ver tarea: #456
Ver commit: #2e342f3
```

[4] Gedit, Atom, Vim, GitHub Desktop

Para hacer etiquetar a un usuario hacerlo de esta manera, esto se puede hacer dentro del cuerpo del mensaje.

Ejemplo:

```
@nombre_usuario
```

Mantén los commit Atómicos

Los commits deberán de ser pequeños, por las siguientes razones:

- Aportaciones por tarea, ya que si en algún momento es necesario revertir, sólo se revierte ese incremento y no afecte a los demás cambios realizados.
- Sólo hacer el commit cuando la tarea este completa.
- Facilidad de juntar el trabajo de los colaboradores.

Bibliografía

Avila, J.-N., & Murzov, I. (2014). *Git*. Obtenido de git-scm: https://git-scm.com/book/en/v2

Beams, C. (31 de Agosto de 2014). *How to Write a Git Commit Message*. Recuperado el 30 de Agosto de 2016, de Chris Beams: http://chris.beams.io/posts/git-commit/

Fherz. (s.f.). *Codigo Facilito*. Recuperado el 30 de Agosto de 2016, de Buenas Practicas en Commits de Git: https://codigofacilito.com/articulos/buenas-practicas-en-commits-de-git