Formato TP Final

Codigo

Tratemos de usar el formato camelCase para nombrar a las variables y a las funciones. Basicamente seria la primer letra en minúscula y cada vez que empieza una nueva palabra se pone en mayúscula (No se usan ‘-’ o ‘\_’).

Por ejemplo: int myNumber;

Las clases, estructuras, enums, tipos de dato, etc tienen que ser escritas siempre con la primer letra mayúscula. En el caso de las clases y estructuras, su nombre debe ser una palabra en singular.

Por ejemplo: class Node{…}; en vez de class nodes{…};

Archivos:

Los archivos deben estar claramente ordenados, sea en la carpeta del proyecto como en el mismo proyecto en visual,

Por ejemplo:

Crear una carpeta con los sprites, otra con muscia y lo mismo con los filtros en el viusal.

Un nuevo cpp y su header pueden ser creados siemrpe que sea una clase nueva o un tipo de función que no tenga relación con las ya existntes.

Commits en Git.

Se debe intentar que todos los commits tengan un mensaje significativo del cambio que se implemento.

Por ultimo, todos los nombres deben estar en ingles y ser significativos, pero no muy redundates.

Hardcodear

Hardcodear es cáncer y nadie quiere cáncer. Hay que reducir los magic numbers al minimo, usando Defines o namespaces . Lo mismo se aplica para strings de texto, que no deberían estar definidos en el código directamente.

Comentarios

Si se agregan printfs o cualquier cosa para debugear, se tiene que marcar con un comentario en esa línea la palabra ‘DEBUG’, asi se encuentra mas rápido cuando lo tengamos listo.

Si se comenta una función entera, se debe escribir dentro de un comentario por que se la comento. Si esa función no sirve mas hay que borrarla.

Se debe comentar que hace cada función y que son los valores que toma y lo que devuelve. Por ejemplo:

// Esta función suma dos números

// - int a: primer numero a sumar.

// -int b: segundo numero a sumar

Int sum (int a, int b);

Allegro

Tenemos un wrapper de Allegro casi completo que incluye desde la inicialización hasta fabricas de displays, primitivas y bitmaps, la idea es usar estas clases en vez de meterse directo con allegro. Vamos a tratar de termianar el wrapper lo antes posible. Tenemos dos tipos de datos especiales, uno es WrittenBox, que es básicamente un cuadro de texto que también se puede transformar en un botón mientras que el otro es WritableBox que seria un cuadro de texto que podes editar.

Hay una clase de allegro que contiene toda la inicialización. La clase esta comentada para que se entender como se usa. Tambien hay un archivo que se llama ‘AllegroAddons’ que no debería usarse por que vamos a usar el modo completo de allego, pero sirve para inicializar las distintas partes de allegro por separado. El ultimo archivo es AllegroFactory, que básicamente es un conjunto de fabricas de objetos de allegro, desde displays hasta bitmaps con distintas características.

https://github.com/ldellisola/Allegro-Template-for-C-

Memoria

En los posible usar Smart pointers en vez de usar new y delete. Lo bueno de los Smart pointers es que evita memory leaks ya que se autodestruye. Hay dos tipos:

* Unique\_ptr: el puntero se autodestruye cuando sale del bloque.
* Shared\_ptr: el puntero se autodestruye cuando se dejan de usar todas las copias de ese puntero.

<https://stackoverflow.com/questions/106508/what-is-a-smart-pointer-and-when-should-i-use-one>