 **Komet Kevin**

**Proyecto Software**

Planificación de coste



Autor: Ramiro Woutersen Uriarte

Fecha: ¿?-¿?-19

***Ingeniería Informática | Curso 2018 - 2019.***



*Komet Kevin por Ramiro Woutersen Uriarte , Sergio Langarita Benitez, Arturo García Enguita, Denis Florin Cobeti se distribuye bajo una*[*licencia de Creative Commons Reconocimiento-NoComercial-SinObraDerivada 4.0 Internacional*](http://creativecommons.org/licenses/by-nc-nd/4.0/)*.*

**Tabla de contenidos**

[5.- Pruebas previstas 3](#__RefHeading___Toc826_1322100828)

[5.1.- Pruebas de unidad 3](#__RefHeading___Toc5030330)

[5.2.- Pruebas de integración 3](#__RefHeading___Toc5030331)

## 5.- Pruebas previstas

Dada la naturaleza gráfica de la aplicación no se podrán realizar demasiadas pruebas de unidad. Para este tipo de software en concreto, un videojuego con su motor, van a ser de especial relevancia las pruebas de integración. Debido a la elevada carga de matemática geométrica es preferible su comprobación a nivel visual que mediante la lectura de variables.

## 5.1.- Pruebas de unidad

De implementarse (existen múltiples librerías geométricas), la única clases sobre la que se pueden realizar pruebas de unidad es Vector2.

**Vector2**

* Tratar de crear, eliminar y editar las dos variables del objeto.
* Realizar cada una de las operaciones (producto escalar, suma de vectores, multiplicar por k...) con vectores con valores nulos.
* Tratar de asignar a un vector un ángulo superior a PI o a -PI

## 5.2.- Pruebas de integración

Las pruebas de integración son incrementales. Hay 4 clases esenciales que han de implementarse previamente a comenzar con las pruebas de integración que son: GameContainer, Image, Window y Renderer. Una vez codificadas y combinadas ya se cuenta con los cimientos de todo el motor gráfico, hay que asegurarse de:

* Correcto dibujado de las formas básicas: punto, recta, circulo y rectángulo
* Correcto dibujado de imágenes (png) con distintos grados de rotación

A partir de este punto son continuas pruebas de integración conforme se añaden clases y se dota de nuevas funcionalidades al motor.

**Input**

* Asegurar la detección de las teclas al presionar, mantener y soltar
* Tomar coordenadas de la Window con el puntero del ratón

**ImageTile**

* Correcto dibujado de imágenes animadas con distintos grados de rotación

**SoundClip**

* Reproducir y pausar sonidos empleando los mecanismos del input (teclas y ratón)

Tras estas pruebas ya se tendría un motor grafico básico y se procedería a la integración de GameContainer, GameObject y todas las clases de colisionadores.

Debido a la complejidad de los códigos de los colisionadores, se recomienda implementar varios hijos de GameObject con pequeñas funcionalidades para probar y asegurar el correcto funcionamiento de BoxCollider, CircleCollider y RayCollider (se puede recurrir a las primitivas de dibujado de renderer para representarlos). Se deberán realizar 3 pruebas asegurando la correcta interacción entre todos ellos.