Desarrollo de Sistemas Distribuidos

Proyecto 3

Elaborado por: Ukranio Coronilla

El proyecto consiste en un simulador de asteroides en dos dimensiones muy sencillo. Este es bastante similar al anterior, sin embargo requiere cumplir los requisitos que se enumeran a continuación, en caso de no cumplir cada uno de ellos, se irá restando calificación.

1.- Para manejar los movimientos de los asteroides vamos a utilizar la abstracción que nos brindan los vectores, de modo que vamos a incluir la siguiente clase Vector2D cuya interfaz se muestra a continuación:

```
#ifndef VECTOR2D_H_
#define VECTOR2D_H_

#include <cmath>

class Vector2D
{
public:
	Vector2D(float, float);
	Vector2D(void);

	float getX();
	float getY();

	void setX(float x);
	void setY(float y);
	float length();
	void normaliza();

	//Sobrecarga para adición de 2 vectores
	Vector2D operator+(const Vector2D& v2) const {
		return Vector2D(m_x + v2.m_x, m_y + v2.m_y);
	}
	friend Vector2D& operator+=(Vector2D& v1, const Vector2D& v2){
		v1.m_x += v2.m_x;
		v1.m_y += v2.m_y;
		return v1;
	}

	//Sobrecara para multiplicar un vector por un escalar
	Vector2D operator*(float scalar){
		return Vector2D(m_x * scalar, m_y * scalar);
	}
```

```
	Vector2D& operator*=(float scalar){
		m_x *= scalar;
		m_y *= scalar;
		return *this;
	}
private:
	float m_x;
	float m_y;
};

#endif
```

La sobrecarga de operadores permite realizar las siguientes operaciones:

```
Vector2D v1(10, 11);

Vector2D v2(35,25);

v1 += v2;

Vector2D v3 = v1 + v2;
```

También permite multiplicar un vector por un escalar de tipo flotante.

Elabore la implementación de la clase Vector2D, donde el constructor sin parámetros inicializa un vector nulo, el método **length** devuelve la magnitud del vector y el método **normaliza** convierte el vector en un vector unitario (normalizado).

2.- La función principal seguirá quedando como lo indica la práctica "*practica_SDL_intro_parte1_ver2.pdf*", y la clase **Game** deberá contener como miembro privado un vector STL de objetos Asteroide:

```
vector <Asteroide> asteroides;
```

La clase Asteroide se explica en el siguiente apartado y sirve para modelar los asteroides.

La ventana para presentar la simulación deberá ser de 800 pixeles de ancho por 600 pixeles de alto.

El constructor de la clase Game recibe el número de asteroides que se van a simular (dicho número se debe pasar como parámetro en la línea de comandos al programa).

El método `void Game::update()` solo se va a encargar de mandar a llamar un método de actualización de posición de todos los objetos Asteroide, así como de borrar los objetos Asteroide (método `erase()`) que ya no se puedan desplegar dentro de la pantalla, así como añadir en este caso nuevos objetos Asteroide mediante el método `emplace_back()` (`push_back()` no es valido en C++11).

El método `void Game::render()` se va a encargar de llamar a las funciones:

SDL_RenderClear, SDL_SetRenderDrawColor, SDL_RenderDrawLines, SDL_SetRenderDrawColor y SDL_RenderPresent.

Observe que para llamar a SDL_RenderDrawLines es necesario para el segundo parámetro que la clase Asteroide devuelva un apuntador al arreglo de `SDL_Point`.

3.- Para la clase Asteroide se deben generar asteroides como objetos irregulares que se desplazan a velocidad constante y van rotando tal y como se muestra en el video. Observe que los objetos más grandes viajan a menor velocidad y rotan a menos revoluciones por minuto.

Un asteroide no debe tener un diámetro mayor de 100 pixeles y el máximo número de vértices debe ser de 20. Es importante que los asteroides sean como los que se muestran en el video, evitando figuras que por su forma sean poco creíbles.

La clase Asteroide debe almacenar los vértices en un vector:

```
vector <Vector2D> vertices;
```

así mismo debe contener un apuntador al arreglo de puntos que puedan ser utilizados por SDL_RenderDrawLines (dicho arreglo debe reservarse de manera dinámica en el constructor):

```
SDL_Point *points;
```

Los asteroides deben entrar o salir por cualquier lado de la pantalla y sus movimientos deben ser continuos.

Para subirse a MOODLE, los códigos de los programas elaborados (principal, interfaz, implementación, Makefile) deben concatenarse en un archivo de texto plano, dejando una línea de asteriscos "" entre cada archivo. Al inicio de cada archivo y en la primera línea debe aparecer el nombre del archivo, posteriormente y en las siguientes líneas un breve comentario sobre lo que realiza dicho archivo. El nombre del archivo debe ser el nombre del alumno separado con guion bajo, materia (DSD), grupo, numero de proyecto y extensión txt. El no cumplir con estos requisitos provocará la disminución de la calificación.*

Ejemplo de un nombre de archivo:

`Juan_Perez_Molinar_DSD_4CM2_2.txt`

Advertencia: Evite copiar programas y que le sean copiados, cualquier acto de plagio se castigará para plagiario y plagiado con cero.