Fundamentos de programación gráfica
Práctica 9 - Sistemas de partículas

En esta práctica, vamos a añadir a nuestra escena soporte para emisores de partículas. Esta práctica está dividida en dos apartados de evaluación independiente.

Primera parte (0,5 puntos)

Debemos añadir al motor las clases `Particle` y `Emitter`. La primera de ellas es una subclase de `Sprite` con la siguiente interfaz:

```
class Particle : public Sprite {
public:
	Particle();
	Particle(Image* image, double velx, double vely, double angularVel, double lifetime, bool autofade);

	virtual double GetLifetime() const;

	virtual void Update(double elapsed);
private:
    double velocityx, velocityy;
    double angularVelocity;
    double lifetime;
    double initialLifetime;
    bool autofade;
};
```

A continuación explicaremos brevemente sus atributos:

- La velocidad de desplazamiento (`velocityx`, `velocityy`) está expresada en puntos por segundo. La partícula debe moverse sobre cada eje a la velocidad indicada.
- La velocidad angular (`angularVelocity`) está expresada en grados por segundo. La partícula debe girar sobre sí misma a esta velocidad.

- El tiempo de vida restante está almacenado en el atributo `lifetime`. El tiempo de vida con el que se creó la partícula se almacena en `initialLifetime`.
- Si el atributo `autofade` es `true`, la partícula debe verse más transparente según se acerca al final de su ciclo de vida.

Respecto a la clase `Emitter`, cuenta con la siguiente interfaz:

```
class Emitter {
public:
	Emitter(Image* image, bool autofade);

	virtual void SetPosition(double x, double y);
	virtual void SetX(double x);
	virtual void SetY(double y);
	virtual double GetX() const;
	virtual double GetY() const;

	virtual void SetRate(double minrate, double maxrate);
	virtual void SetVelocityX(double minvelx, double maxvelx);
	virtual void SetVelocityY(double minvely, double maxvely);
	virtual void SetAngularVelocity(double minangvel, double maxangvel);
	virtual void SetLifetime(double minlifetime, double maxlifetime);
	virtual void SetMinColor(uint8 r, uint8 g, uint8 b);
	virtual void SetMaxColor(uint8 r, uint8 g, uint8 b);
	virtual void SetBlendMode(Renderer::BlendMode mode);

	virtual void Start();
	virtual void Stop();
	virtual bool IsEmitting() const;

	virtual void Update(double elapsed);
	virtual void Render() const;
private:
	Image* image;
	bool autofade;
	double x, y;

	double minrate, maxrate;
	double minvelx, maxvelx;
	double minvely, maxvely;
	double minangvel, maxangvel;
	double minlifetime, maxlifetime;
	uint8 minr, ming, minb;
	uint8 maxr, maxg, maxb;
	Renderer::BlendMode blendMode;

	bool emitting;
```

```
        Array<Particle> particles;
};
```

Los atributos de la clase son los siguientes:

- Una imagen para las partículas.
- Auto fundido para las partículas.
- Posición del emisor.
- Tasa de emisión mínima y máxima (cuántas partículas se emiten por segundo).
- Velocidad en X e Y mínima y máxima.
- Velocidad angular mínima y máxima.
- Valores de color mínimos y máximos.
- Modo de pintado de las partículas (por defecto debe ser `ADDITIVE`).
- Un atributo indicando si el emisor está generando partículas en ese momento o no.
- Una lista con las partículas existentes.

A la hora de emitir partículas, para todos los valores con rango mínimo y máximo, debemos generar un número aleatorio dentro de ese rango para cada partícula.

Una vez implementadas ambas clases, se deben incluir sus cabeceras en los ficheros que sea necesario, y en la clase `Scene`, descomentar las líneas relativas a la gestión de los emisores de partículas.

Realizaremos un ejercicio para probar la implementación de las partículas. En dicho ejercicio, debemos crear una escena de fondo negro con un sprite que utilice la imagen **“data/star.png”** que siga al puntero del ratón.

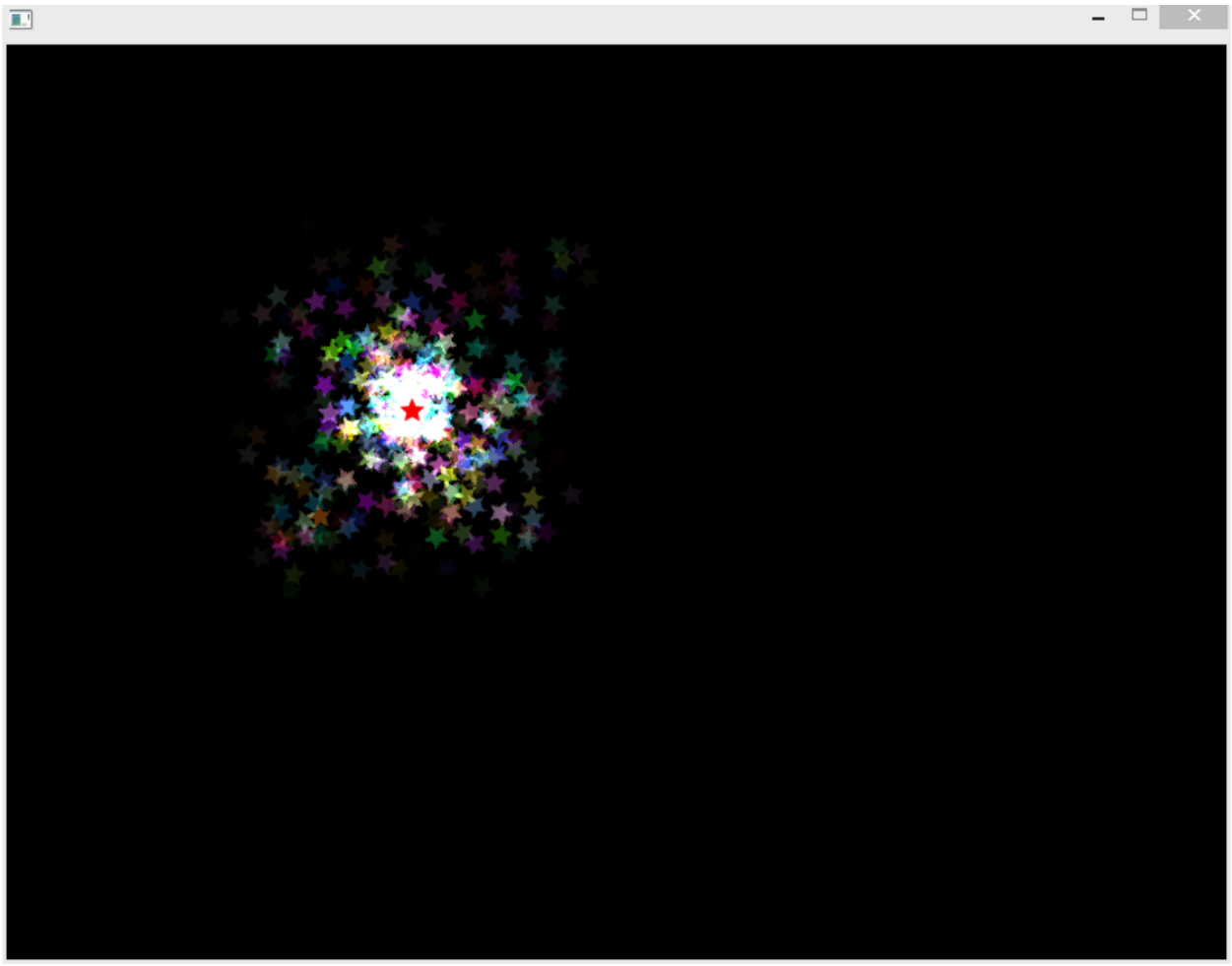
Debe haber un emisor de partículas que también siga al puntero del ratón. Dicho emisor debe tener los siguientes rangos para las partículas que genere:

- Tasa de emisión: 500, 1000
- Velocidad en X: -128, 128
- Velocidad en Y: -128, 128
- Velocidad angular: 0, 360
- Tiempo de vida: 1, 2
- Valores mínimos de color: 0, 0, 0
- Valores máximos de color: 255, 255, 255

Cuando una partícula llega al final de su tiempo de vida (es decir, cuando su método `GetLifetime` devuelve 0), debe ser eliminada por el emisor.

Cuando se pulse el botón izquierdo del ratón, el emisor debe comenzar a emitir partículas. Si el botón izquierdo no está pulsado, debe cesar la emisión de partículas.

A continuación una imagen mostrando el aspecto que tendrá el ejercicio si está correctamente resuelto:



Segunda parte (0,5 puntos)

En esta segunda parte, vamos a añadir afectores básicos a nuestro motor. Estos afectores modificarán los atributos de todas las partículas que pasen por una determinada región de la escena.

Se debe definir la interfaz e implementación de una clase `Affector` con las siguientes características:

- Región de la escena en la que se aplica el afector.
- Nuevos rangos de colores.
- Nuevos rangos de velocidad de desplazamiento.
- Nuevo rango de velocidad angular.

Se podrán añadir estos afectores a un emisor, y afectarán a todas las partículas del emisor que entren en la región de la escena especificada por el afector. Un afector debe afectar una única vez a cada partícula, así que deberá almacenar una lista de partículas que ya ha modificado

para no volver a hacerlo. Cuando una partícula es eliminada por el emisor, también debe borrarse de la lista de partículas afectadas por un determinado afector.

Para probar la implementación de afectores, modificaremos el ejercicio anterior añadiendo dos afectores, uno ocupando la mitad izquierda de la pantalla, y otro ocupando la mitad derecha.

El afector de la izquierda debe tener los siguientes rangos:

- Valores mínimos de color: 0, 0, 0
- Valores máximos de color: 255, 255, 0
- Velocidad angular: 0, 360

El afector de la derecha debe tener los siguientes rangos:

- Valores mínimos de color: 0, 0, 0
- Valores máximos de color: 0, 255, 255
- Velocidad angular: 360, 720