Tecnología de la Programación

Proyecto de Programación

Curso 2015-2016

- Alumno: Bogach Yurievich, Kyryl.
- Grupo: 2
- Subgrupo: 2

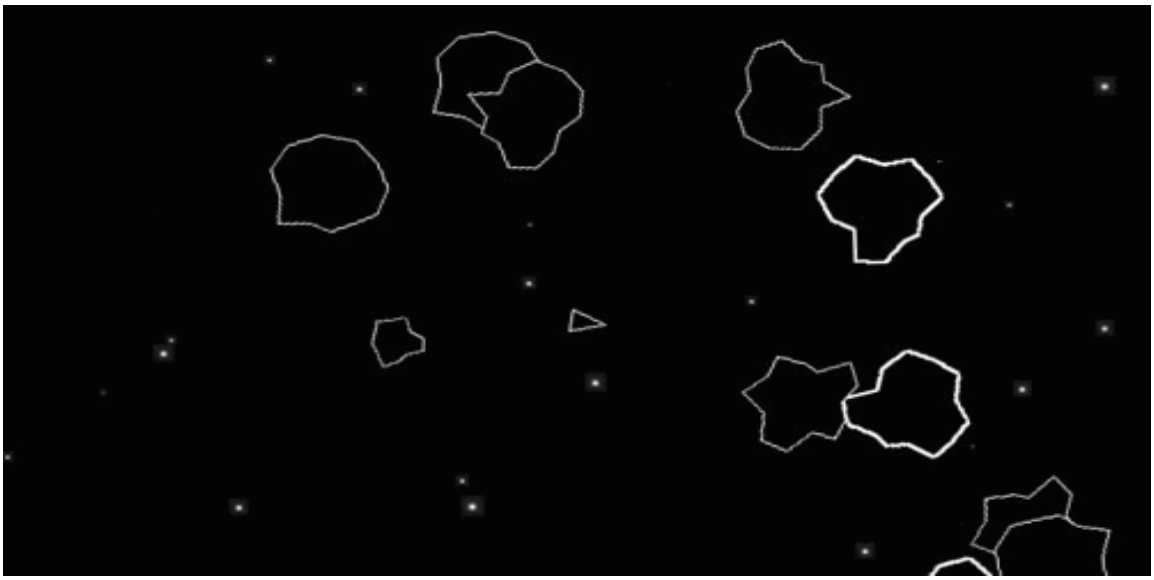
- Convocatoria: mayo 2016

- Profesora: Gracia Sánchez Carpena.

TABLA DE CONTENIDOS

Descripción de la aplicación.

La aplicación desarrollada es un videojuego inspirado en la empresa *Atari* y su juego "Asteroids".

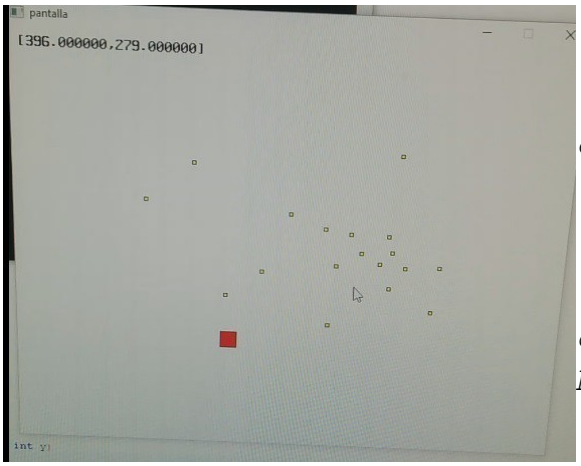


Mi objetivo principal ha sido la creación de un juego *arcade* que se parezca al mencionado pero con un toque personal centrándome en:

- Buscar la forma más eficiente de hacer que me sigan los asteroides.
- Buscar una lógica eficiente para colisiones.
- Un diseño más actual que el de "Atari".
- Una estructura dividida en fases bien diferenciadas.
- Aplicación de conocimientos vectoriales adquiridos en la asignatura de AMD.
- Investigación del código RGB y sus derivados.
- Aritmética modular para el control de estados del menú.
- Sistema de ficheros para guardas puntuaciones.

El desarrollo del proyecto se ha dividido en **3 partes:**

1°: Esqueleto básico del videojuego.



Utilización de cuadrados como elementos del videojuego.

Utilización de la dirección (x,y) del ratón para centrar elementos posteriores.

2°: Implementación del menú.

Imágenes para texto más grande.

Fondo con animación.

Record en pantalla.

3°: Implementación de imágenes y detalles.

Con sistema de puntuación.

Con sistema de vidas.

Fin de juego.

Guardar puntuación.

Reseteo de partida.

Manual de usuario.

- Introducir nombre del jugador.
- Seleccionar la opción del menú que se desea ya se bien, por las flechas del teclado y dándole Intro, o bien haciendo clic izquierdo del ratón en la opción.
- Muévete con las teclas WASD del teclado.
- Disparar con el clic izquierdo del ratón (único disparo).
- Disparar con el clic derecho del ratón (disparo cruzado).
- Hay un límite de balas explosivas; se regenera con el tiempo.
- Esquivar o disparar a los asteroides marrones y los asteroides lilas (te siguen).
- Por cada asteroide marrón eliminado aparecerá un punto verde que te seguirá y deberás absorber para ganar puntuación.
- ¡Cuidado! Si lo matas te restará un punto.
- Por cada asteroide lila eliminado se te proporcionará con 1/3 de corazón.
- Matando 3 asteroides lilas ganarás una vida.
- Pulse Y al terminar la partida para guardar partida.
- Pulse R para reiniciar la partida.
- El jugador con más puntuación estará en el *record*.

Organización del proyecto.

Funcionalidad vectorial en el videojuego:

La principal funcionalidad de este videojuego es el movimiento de los enemigos y de los disparos:

Aplico vectores directores (x,y) para controlar así la trayectoria de todos mis elementos.

Hay que tener en cuenta un factor muy importante que es el de utilizar vectores unitarios. ¿Por qué? Debemos utilizarlos para crear una uniformidad entre los elementos de un tipo de personajes, como por ejemplo:

Los **disparos**, se crea el vector director (cuando se haga clic en la pantalla) entre el puntero del ratón y la posición de la nave. Éste disparo se moverá en función de ese vector y si no se convierte en unitario el disparo irá más rápido si hago clic más lejos de la nave que si se hace más cerca. Así mismo, se aplica el vector unitario y se multiplica por la velocidad.

Los **asteroides marrones** toman la dirección de la posición de la nave al crearse. Se toma el vector unitario y se multiplica por la velocidad.

Los **asteroides lilas y los puntos** toman la dirección de la posición de la nave al crearse. Se toma el vector unitario y se multiplica por la velocidad. En este caso, la dirección se actualiza en cada iteración del bucle principal creándose así una especie de "seguimiento" de los asteroides lilas con la nave.

$$\mathbf{u} = \frac{\mathbf{v}}{|\mathbf{v}|} = \frac{1}{|\mathbf{v}|}\mathbf{v}$$

Es importante en todos los casos anteriores tener en cuenta que los vectores tienen **sentido** y por lo tanto se debe tomar un orden concreto.

Para las colisiones de los rectángulos:

Para que un rectángulo esté colisionando con otro deben cumplirse cuatro cosas, definiendo dos rectángulos como r1 y r2 diremos que colisionan si:

1. Lado derecho de r1 es mayor que lado izquierdo de r2
2. Lado izquierdo de r1 es menor que lado derecho de r2
3. Lado superior de r1 es mayor que lado inferior de r2
4. Lado inferior de r1 es menor que lado superior de r2

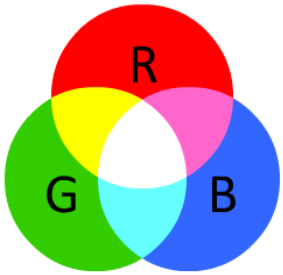
Si se cumplen estas cuatro condiciones es que ambos rectángulos están colisionando.

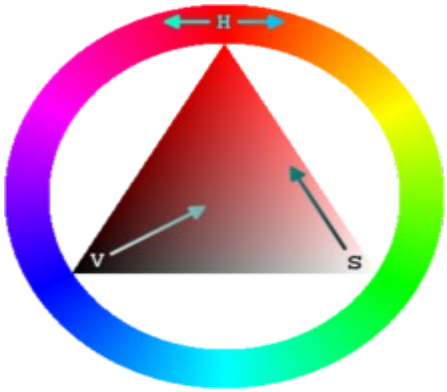
Información de: http://www.genbetadev.com/

Para el recorrido de las tonalidades en función de una variable:

He utilizado una alternativa al código RGB:

RGB(lineal) → HSV (vectorial)

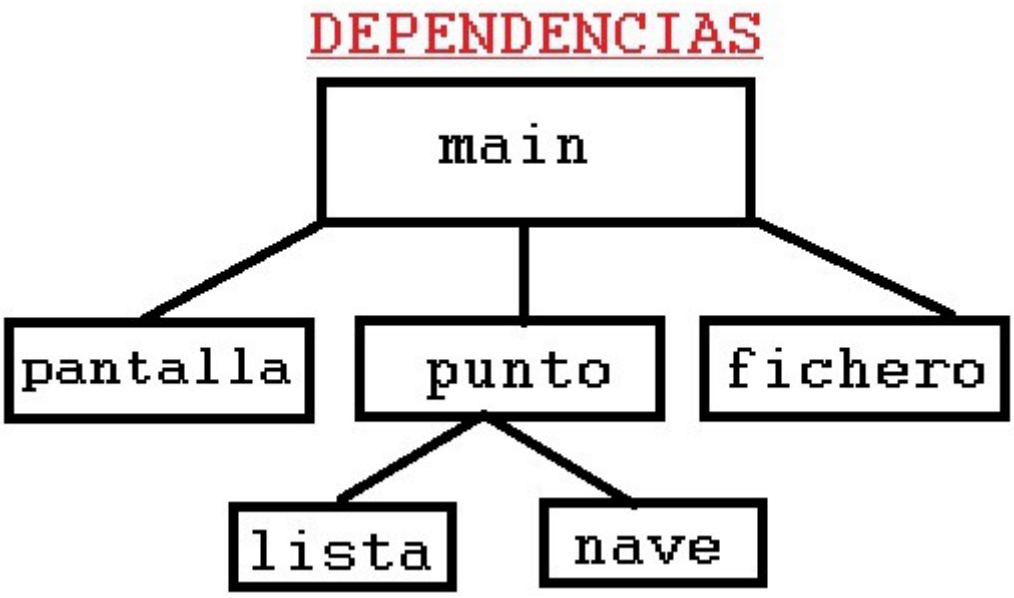


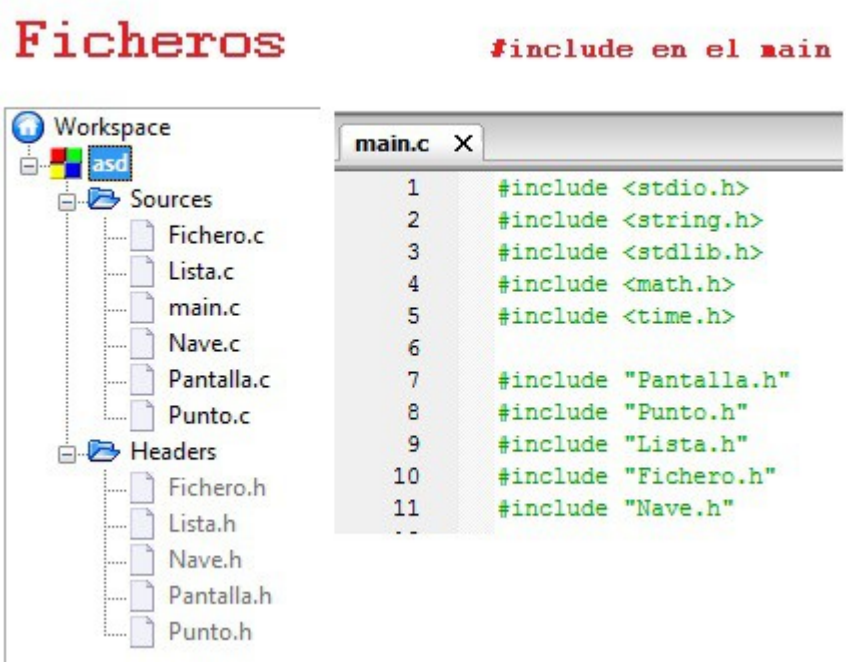


Utilizando el código de ésta página:

https://www.cs.rit.edu

Dependencias entre los módulos y ficheros que componen el proyecto.





Estructura de datos.

TDA Simples: Punto y Nave.

TDA Punto facilita durante la programación del proyecto la utilización de las coordenadas (x,y) en pantalla de los elementos del juego.
TDA Nave utiliza como base un Punto y permite el control de disparos y/o movimientos de la Nave.

TDA Contenedores: Lista y Fichero.

TDA Lista presenta una estructura de datos enlazada de simple enlace con cabecera para facilitar la inserción, creación y control de los elementos del videojuego como los enemigos o las balas.
TDA Fichero presenta una estructura de datos enlazada representado internamente con arrays para hacer la gestión de ficheros y poder, de este modo, guardar las puntuaciones y el nombre de los jugadores.

Conclusiones.

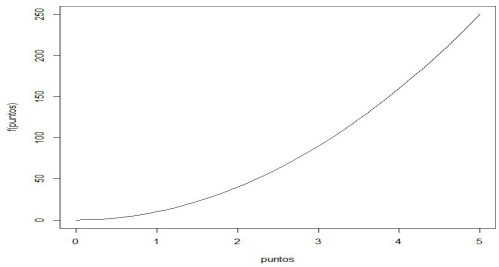
N°1: Una de las dificultades que se me han presentado a lo largo de la programación de éste proyecto ha sido el del recorrido de dos listas:

```
int ListaColisiones (Lista p, Lista r, double plargo, double palto, double qlargo, double qalto) ///balas - enemigo
{
    int contador=0;
    while (p->sig!=NULL)
    {
        Lista q=r;
        while (q->sig!=NULL)
        {
            double rect1_x=PuntoX(p->sig->pos);
            double rect2_x=PuntoX(q->sig->pos);
            double rect1_y=PuntoY(p->sig->pos);
            double rect2_y=PuntoY(q->sig->pos);
            if ( rect1_x < rect2_x + qlargo
                    && rect1_x + plargo > rect2_x
                    && rect1_y < rect2_y + qalto
                    && rect1_y + palto > rect2_y )
            {
                ListaNodoLibera(q);
                contador++;
            }

         else q=q->sig;

        }
        p=p->sig;
    }
    return contador;
}
```

Debemos resetear a la cabezera la segunda lista cuando la hemos recorrido una vez.

Como resultado, únicamente se me comprobaba la colisión de la primera bala. Tras añadir *Lista q=r* (el reseteo), las colisiones fueron perfectamente.

N°2: En cuanto al recorrido de colores de la barra de munición en función de una variable (azul → rojo), se me ocurrió en primer lugar aplicar algunas de las gráficos que veíamos en *Estadística*. De este modo, encontré que debía ajustar dos gráficas (una y su inversa) para hacer el graduado de colores. Funcionaba bien pero me resultó más atractiva la opción del código HSV.




```
void BarraMunicion(Nave p, double *x){

    int azul=pow((*x),2)*10;
    int verde=azul-50; if (verde<0) verde=0;
    Pantalla_ColorRelleno(255-azul,verde,azul,255);
    Punto q=NavePos(p);
    Pantalla_DibujaRectangulo(PuntoX(q),PuntoY(q)+35,NaveLargo(p)/5*(*x),5);
    if (*x<5.0) *x=*x+0.02;
}
```

Opinión personal:

El grado de satisfacción con ésta asignatura ha sido, por mi parte, muy positiva. Se nos ha permitido realizar un proyecto a nuestro gusto lo cual es muy bueno para incentivar el amor por la programación.

Sin embargo, no todos los alumnos han visto este proyecto como una oportunidad para aprender a base de realizar un juego. Se ha visto un poco como "*lo quiero terminar ya mismo y entregarlo*". Existía una especie de "*temor*" por preguntar las dudas al profesor, por lo cual se acudía a personas "*que sabían*" o que tenían el proyecto medio terminado.

Por mi parte, al ponerme por mi cuenta antes del tema 1 hice todo el videojuego en el main (muy mal por mi parte). De este modo, al ver realmente que las estructuras de TDA son muy útiles para modular el proyecto ves exactamente como fluye el proyecto.

Además, la implementación de estructuras de simple enlace...etc, sirve como base e introducción a la programación por objetos (imprescindible, por lo tanto, absorber todo de ésta asignatura).