

**INSTITUTO TECNOLÓGICO DE IGUALA**

INGENIERÍA EN SISTEMAS COMPUTACIONALES

MATERIA: GRAFICACIÓN

PROFESOR: ARTURO CARLOS RODRÍGUEZ ROMÁN

INTEGRANTES:

* GERARDO SÁNCHEZ ROMERO
* GUADALUPE GARCIA CARDENAS

INFORME DE JUEGO DE ASTEROIDES

TURNO: VESPERTINO

PERIODO: ENERO-JUNIO 2018

GRUPO:” B” AULA: E-3

FECHA:30/05/2018

**INTRODUCCION**

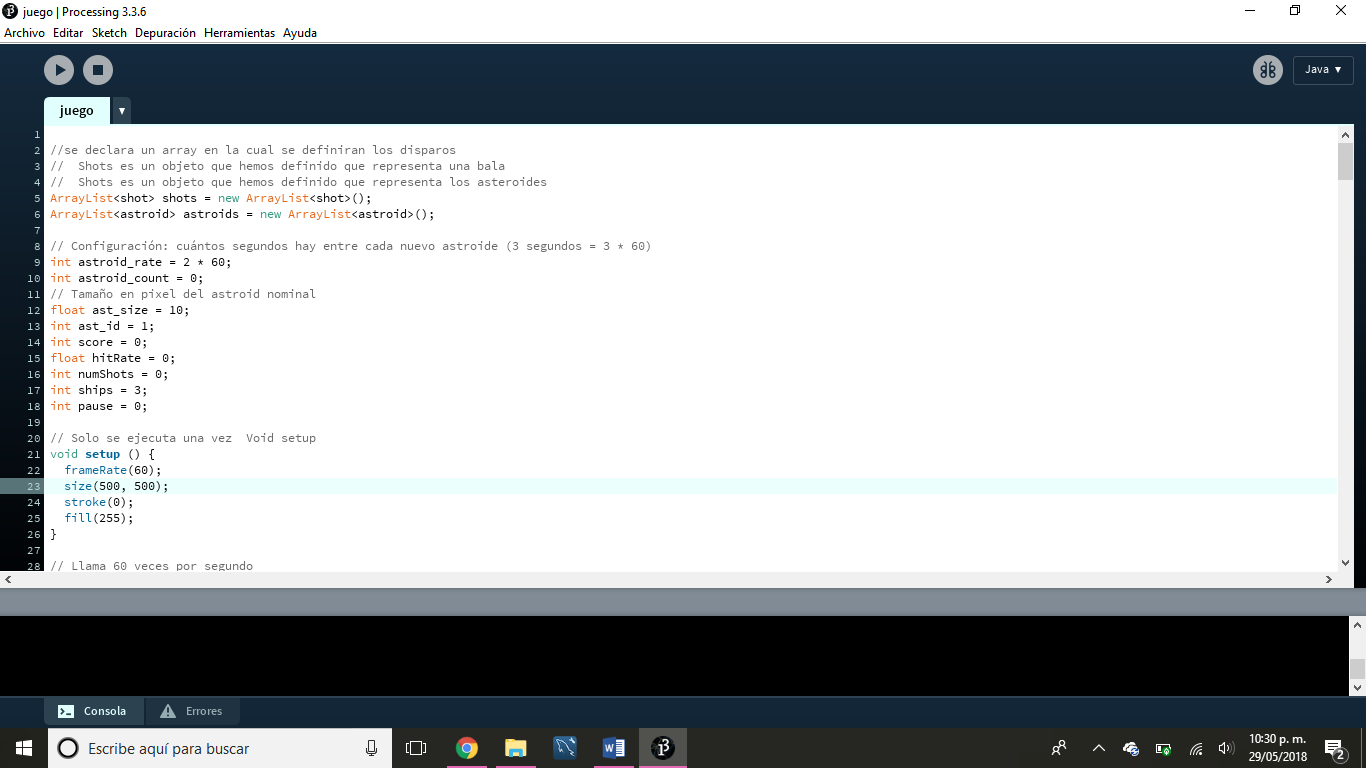
Se realizó la investigación de un juego en el cual se implementó todo lo visto en el curso de graficación, en dicho curso se trabajó en Procesing que es un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, de fácil utilización, y que sirve como medio para la enseñanza y producción de proyectos multimedia e interactivos de diseño digital. Uno de los objetivos declarados de Processing es el de actuar como herramienta para que artistas, diseñadores visuales y miembros de otras comunidades ajenos al lenguaje de la programación, aprendieran las bases de la misma a través de una muestra gráfica instantánea y visual de la información. En este informe se realizará el proceso de análisis y se pretende explicar los procesos que se realizan en el código del juego que se investigó y hacía explicar algunas funciones para hacer del código más fácil de entender.

**Juego de asteroides**

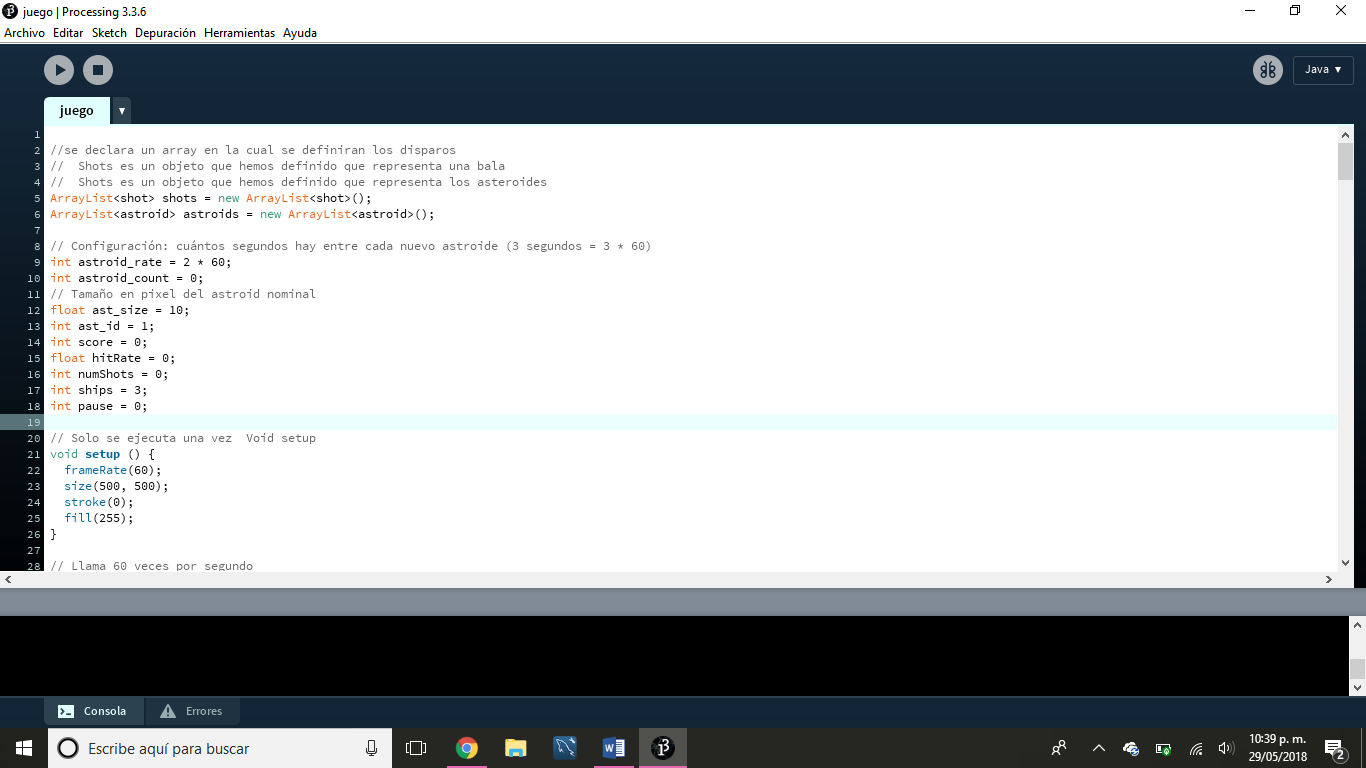
1. Primero que nada, el juego se llama juego.

**Nota: En cada imagen se muestra lo que hace cada proceso.**

Un **ArrayList** almacena una cantidad variable de objetos. Esto es similar a crear una matriz de objetos, pero con ArrayList, los elementos pueden agregarse y eliminarse fácilmente de ArrayList y su tamaño se cambia dinámicamente.



1. Definimos el tipo de variables que utilizaremos para la creación de un nuevo asteroide y en su defecto el tamaño en nuestro caso utilizamos el tipo de variables **int** que son enteros.



1. En este proceso definimos el tamaño, el color utilizado para rellenar formas y color utilizado para dibujar líneas y bordes alrededor de formas.

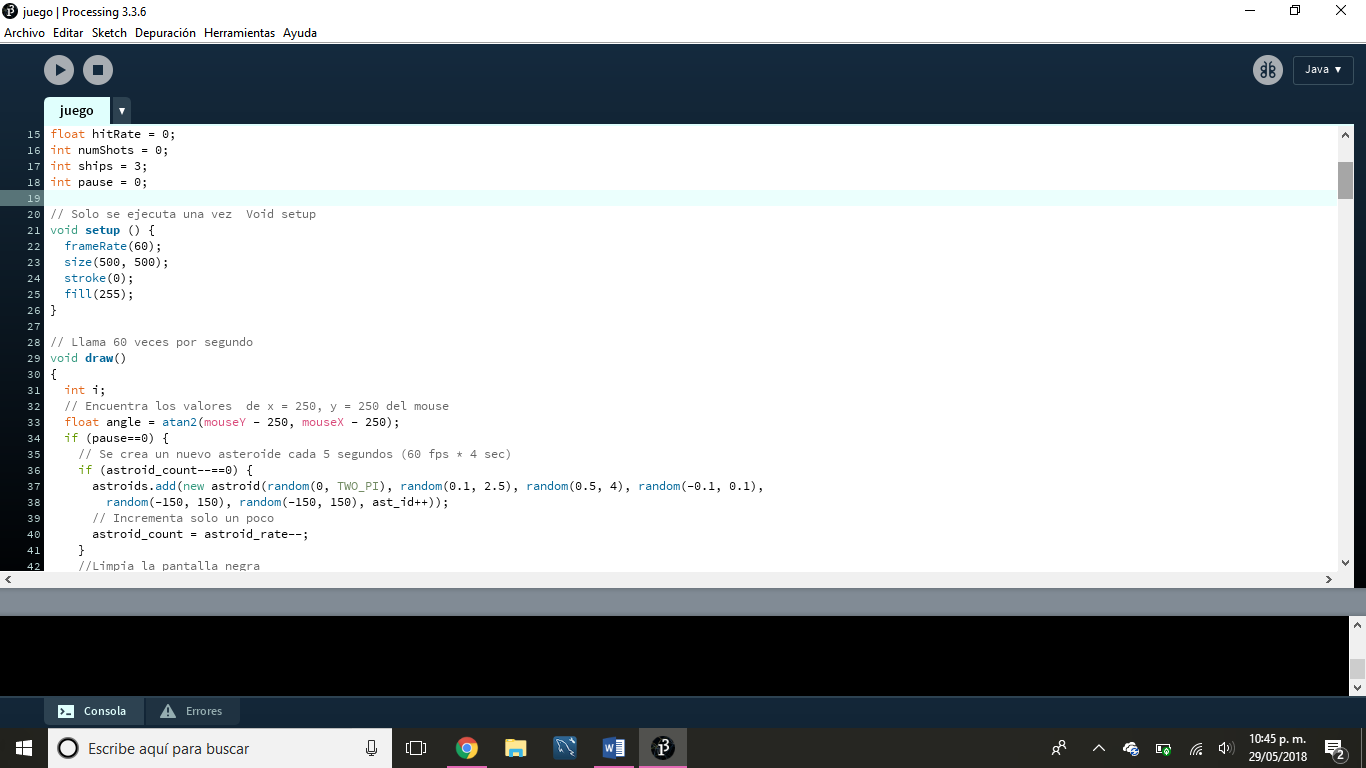
**Funciones que se utilizaron:**

**frameRate:** Especifica la cantidad de cuadros que se mostrarán cada segundo. Por ejemplo, la función llamada frameRate (30) intentará actualizar 30 veces por segundo.

**size:** Define la dimensión del ancho y alto de la ventana de visualización en unidades de píxeles. En un programa que tiene la función setup (), la función size () debe ser la primera línea de código dentro de setup ().

**Stroke:** Establece el color utilizado para dibujar líneas y bordes alrededor de formas. Este color se especifica en términos del color RGB o HSB dependiendo del colorMode actual ().

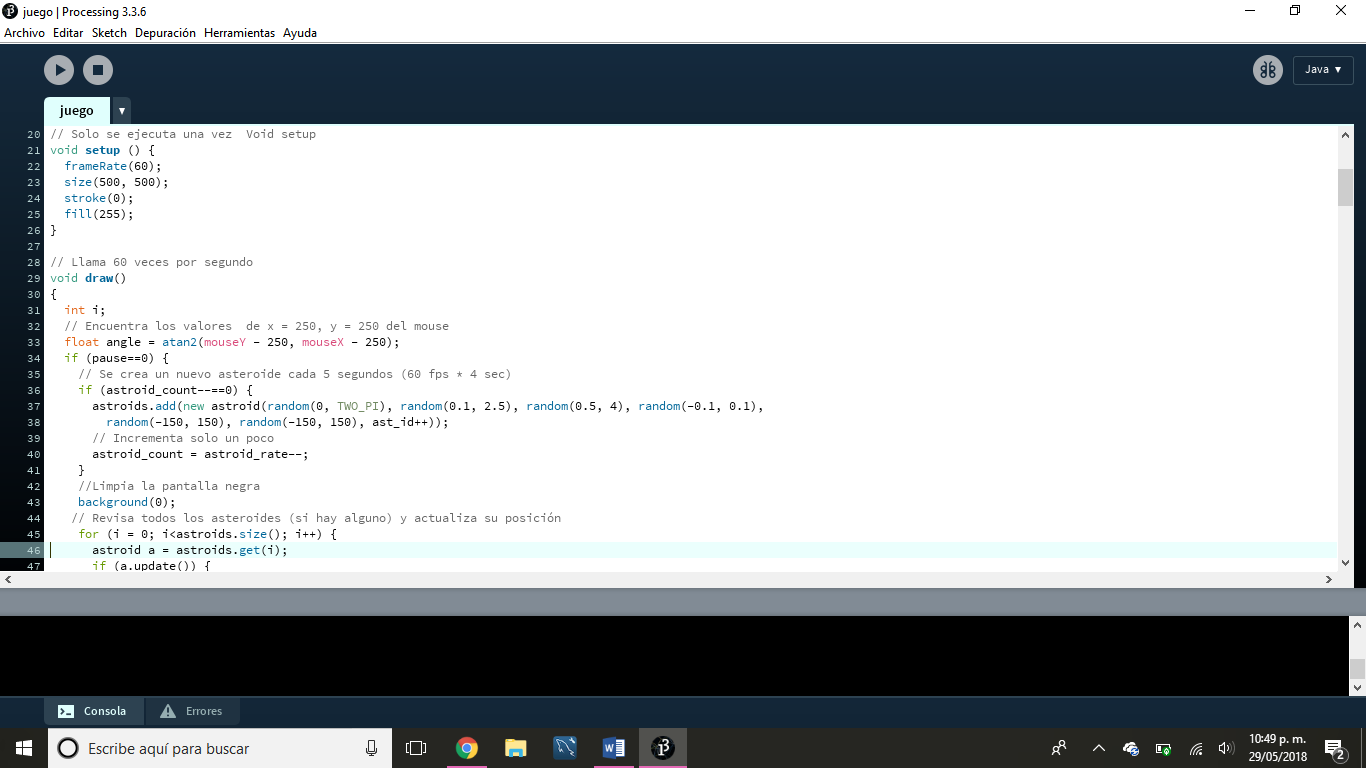
**Fill:** Establece el color utilizado para rellenar formas. Por ejemplo, si ejecuta relleno (204, 102, 0), todas las formas siguientes se llenarán de naranja. Este color se especifica en términos del color RGB o HSB dependiendo del colorMode actual ().



1. En este proceso actualiza los valores 60 veces por segundo.

**Funciones que se utilizaron:**

**Random:** Genera números aleatorios. Cada vez que se llama a la función random (), devuelve un valor inesperado dentro del rango especificado. Si solo se pasa un parámetro a la función, devolverá un flotante entre cero y el valor del parámetro alto.



1. En este proceso se pone el fondo en negro con la función background, revisamos la posición de los asteroides y si hay alguno se actualiza, el for lo ocupamos para hacer un ciclo y verificar tamaños, al igual verificamos si la bala aún sigue en la interfaz y si no sigue la bala se remueve. Él comando if lo ocupamos en este proceso para verificar una colisión de la nave con un asteroide.

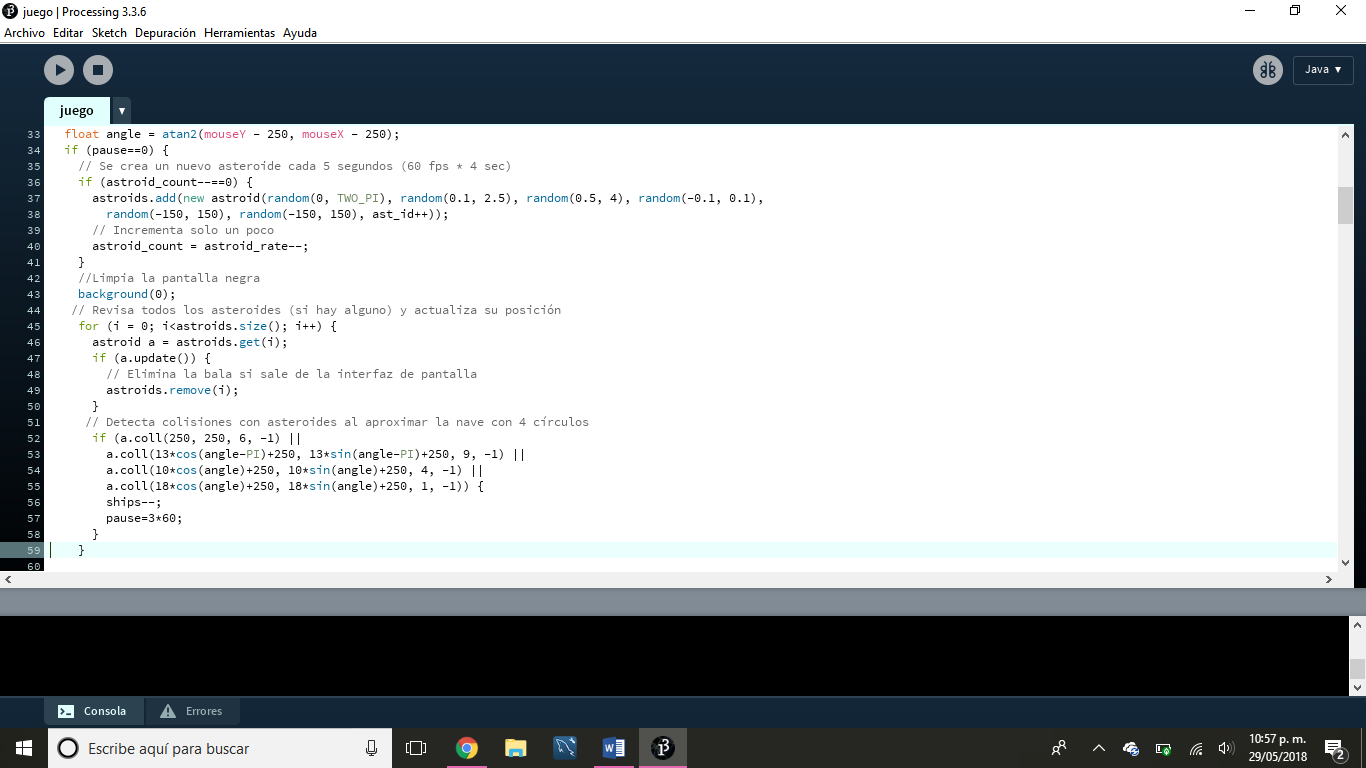
**Funciones que se utilizaron:**

**Background:** La función background () establece el color utilizado para el fondo de la ventana de procesamiento. El fondo predeterminado es gris claro. Esta función se usa típicamente en draw () para borrar la ventana de visualización al comienzo de cada cuadro, pero se puede usar dentro de setup () para establecer el fondo en el primer cuadro de animación o si el fondo solo debe establecerse una vez.

**Get:** Lee el color de cualquier píxel o toma una sección de una imagen. Si no se especifican parámetros, se devuelve la imagen completa. Use los parámetros x e y para obtener el valor de un píxel. Obtenga una sección de la ventana de visualización especificando parámetros w y h adicionales. Al obtener una imagen, los parámetros x e y definen las coordenadas para la esquina superior izquierda de la imagen, independientemente del modo de imagen actual ().

**Sin:** Calcula el seno de un ángulo. Esta función espera que los valores del parámetro de ángulo se proporcionen en radianes (valores de 0 a 6.28). Los valores se devuelven en el rango de -1 a 1.

**Cos:** Calcula el coseno de un ángulo. Esta función espera que los valores del parámetro de ángulo se proporcionen en radianes (valores de 0 a PI \* 2). Los valores se devuelven en el rango de -1 a 1.



1. En este proceso guardamos el estado actual, la rotación que se asignará y la creación del triángulo que será nuestra nave

**Funciones que se utilizaron:**

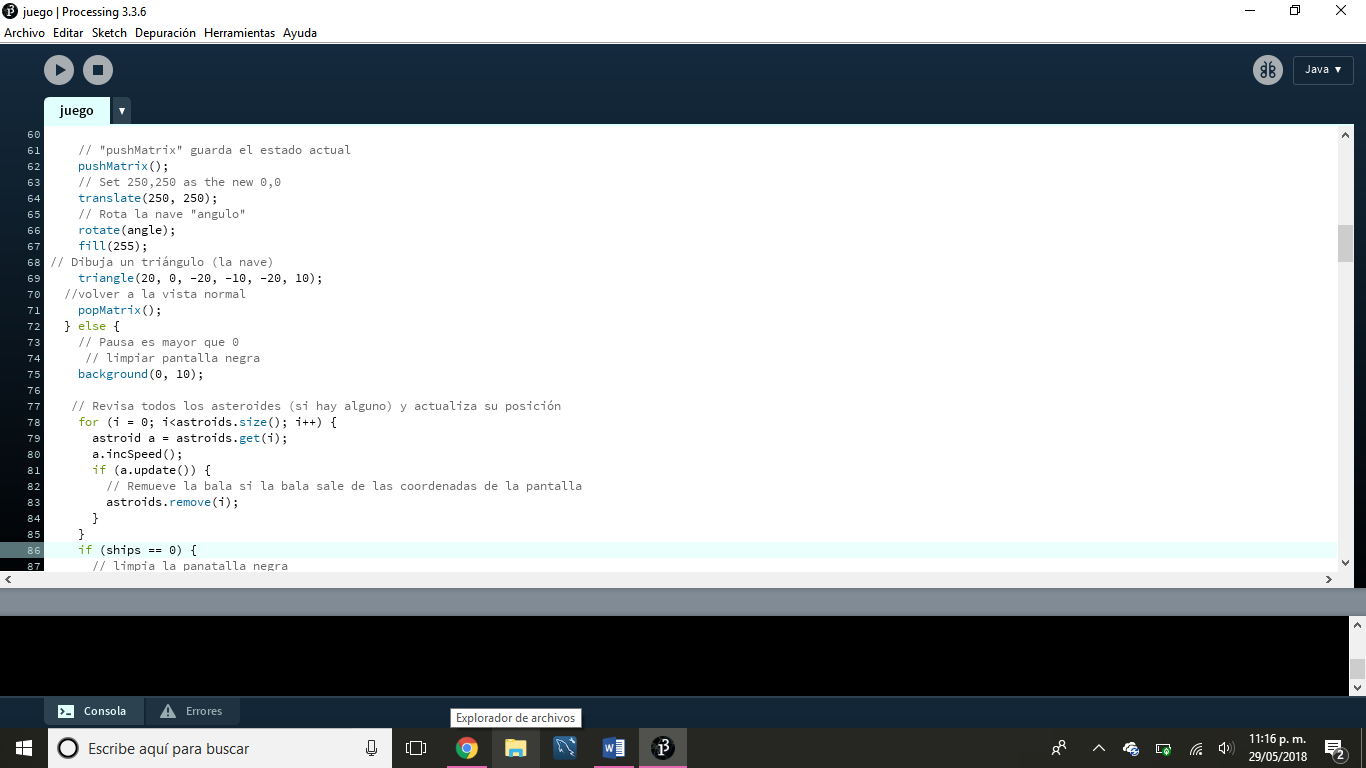
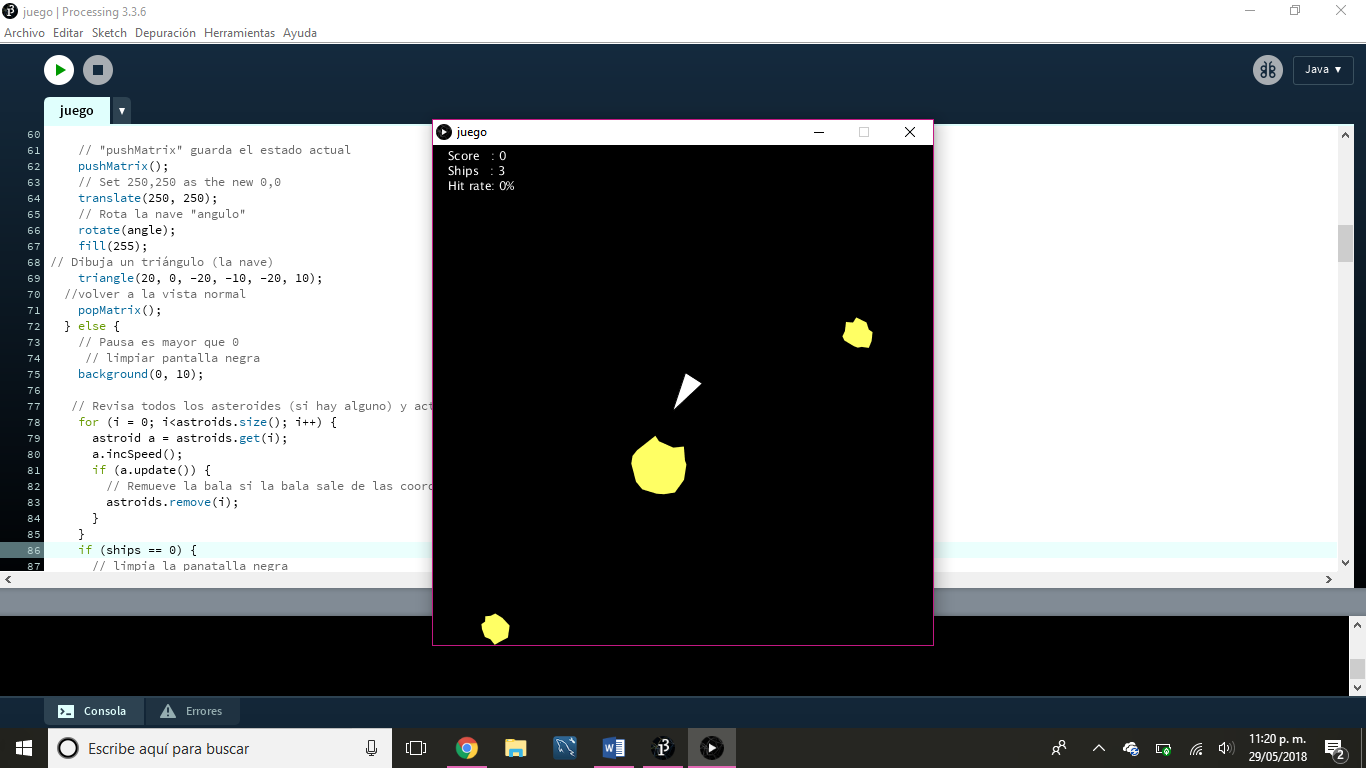
**pushMatrix:** Impulsa la matriz de transformación actual a la pila de la matriz. Comprender pushMatrix () y popMatrix () requiere entender el concepto de una pila de matriz. La función pushMatrix () guarda el sistema de coordenadas actual en la pila y popMatrix () restaura el sistema de coordenadas anterior.

**translate:** Especifica una cantidad para desplazar objetos dentro de la ventana de visualización. El parámetro x especifica la traducción izquierda / derecha, el parámetro y especifica la conversión ascendente / descendente, y el parámetro z especifica las traducciones hacia / desde la pantalla. El uso de esta función con el parámetro z requiere el uso de P3D como parámetro en combinación con el tamaño como se muestra en el ejemplo anterior.

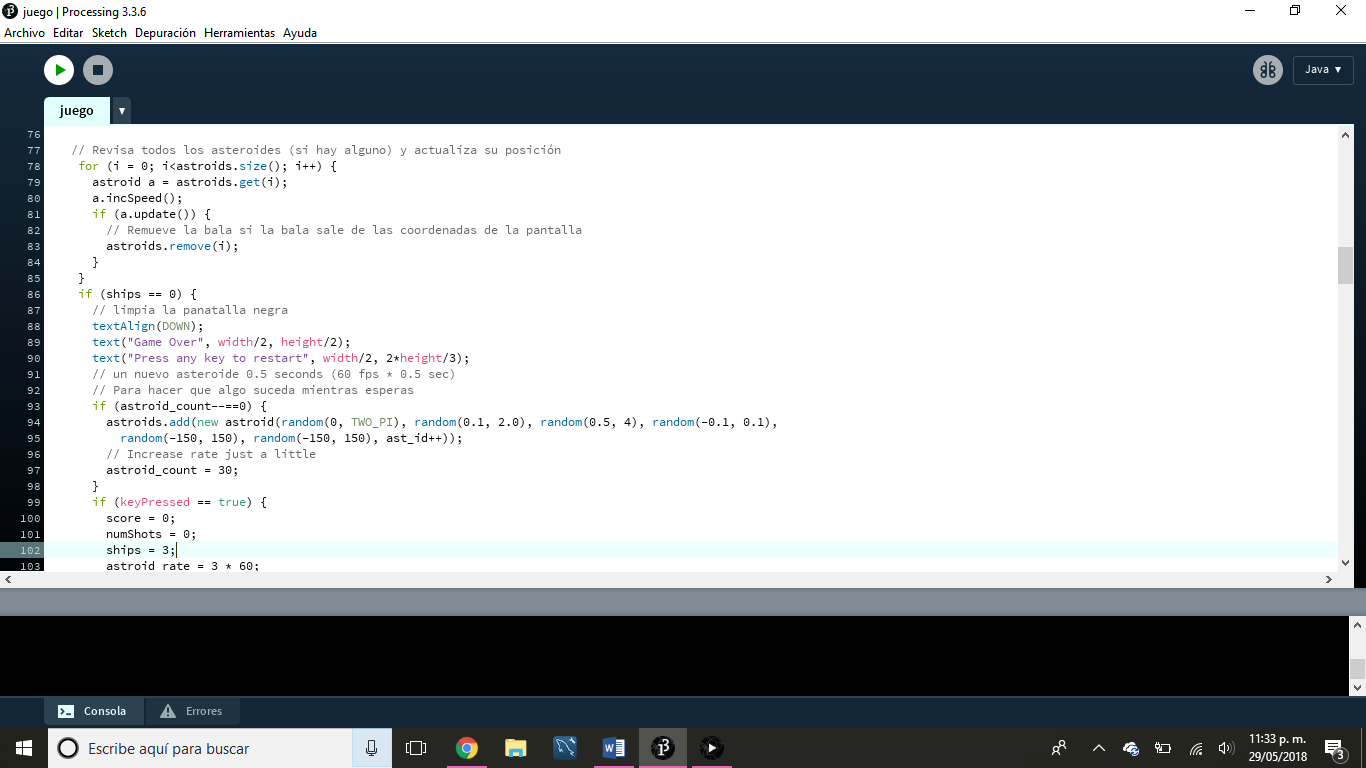
**rotate:** Gira la cantidad especificada por el parámetro de ángulo. Los ángulos se deben especificar en radianes (valores de 0 a TWO\_PI), o se pueden convertir de grados a radianes con la función radianes ()

**triangle:** Un triángulo es un plano creado al conectar tres puntos. Los primeros dos argumentos especifican el primer punto, los dos medios dos especifican el segundo y los dos últimos especifican el tercero.

**popMatrix:** Coloca la matriz de transformación actual fuera de la pila de la matriz. Comprender cómo presionar y explotar requiere comprender el concepto de una pila de matriz. La función pushMatrix () guarda el sistema de coordenadas actual en la pila y popMatrix () restaura el sistema de coordenadas anterior.

1. Revisamos la posición de asteroides y actualizamos.



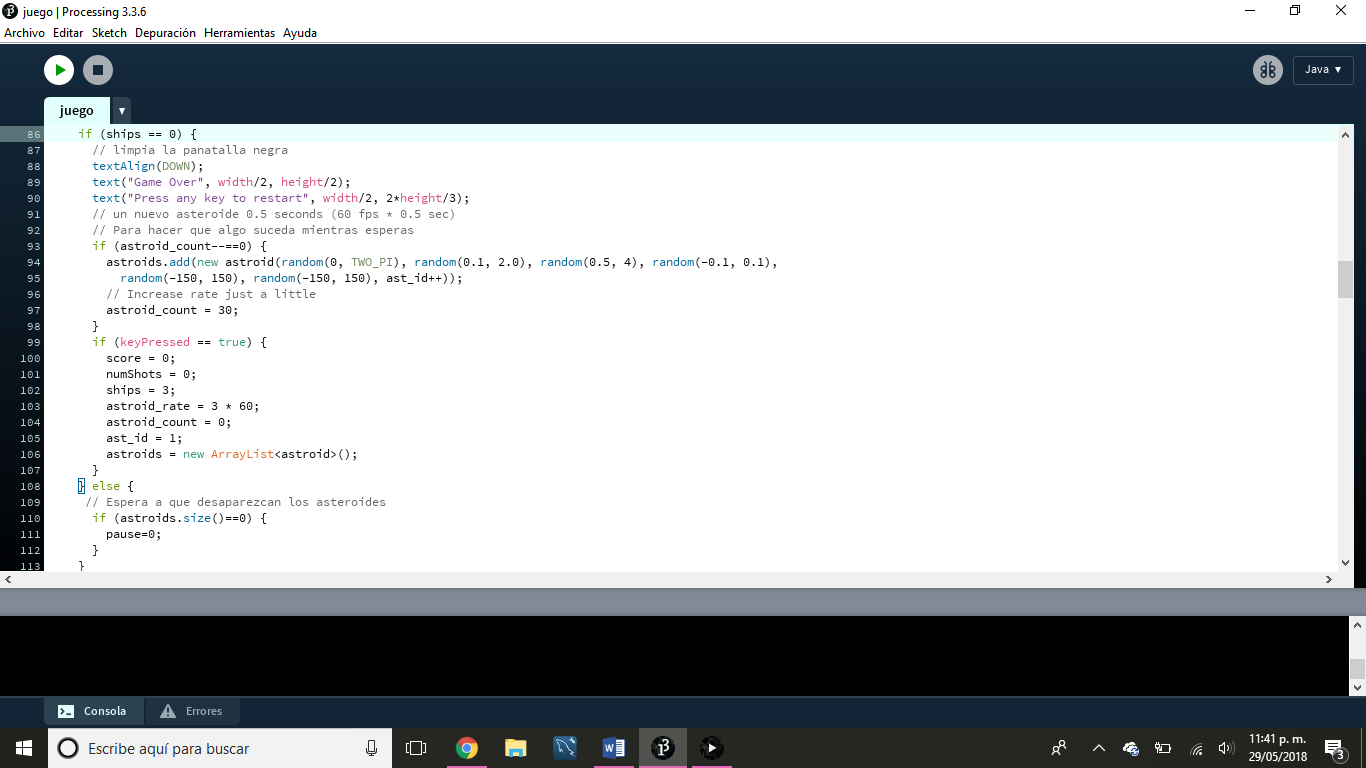
1. En este apartado se muestra que si se impacta 3 veces y el valor decrementa a cero se nos mostrara “Press any key to restart” que al dar click nuevamente en la pantalla se vuelva a iniciar el juego

**Funciones que se utilizaron:**

**textAlign:** Establece la alineación actual para dibujar texto. Los parámetros IZQUIERDA, CENTRAL y DERECHA establecen las características de visualización de las letras en relación con los valores para los parámetros x e y de la función de texto ().

**Random:** Genera números aleatorios. Cada vez que se llama a la función random (), devuelve un valor inesperado dentro del rango especificado. Si solo se pasa un parámetro a la función, devolverá un flotante entre cero y el valor del parámetro alto.

**La función keyPressed ()** se invoca una vez cada vez que se presiona una tecla. La tecla presionada se almacena en la variable clave.



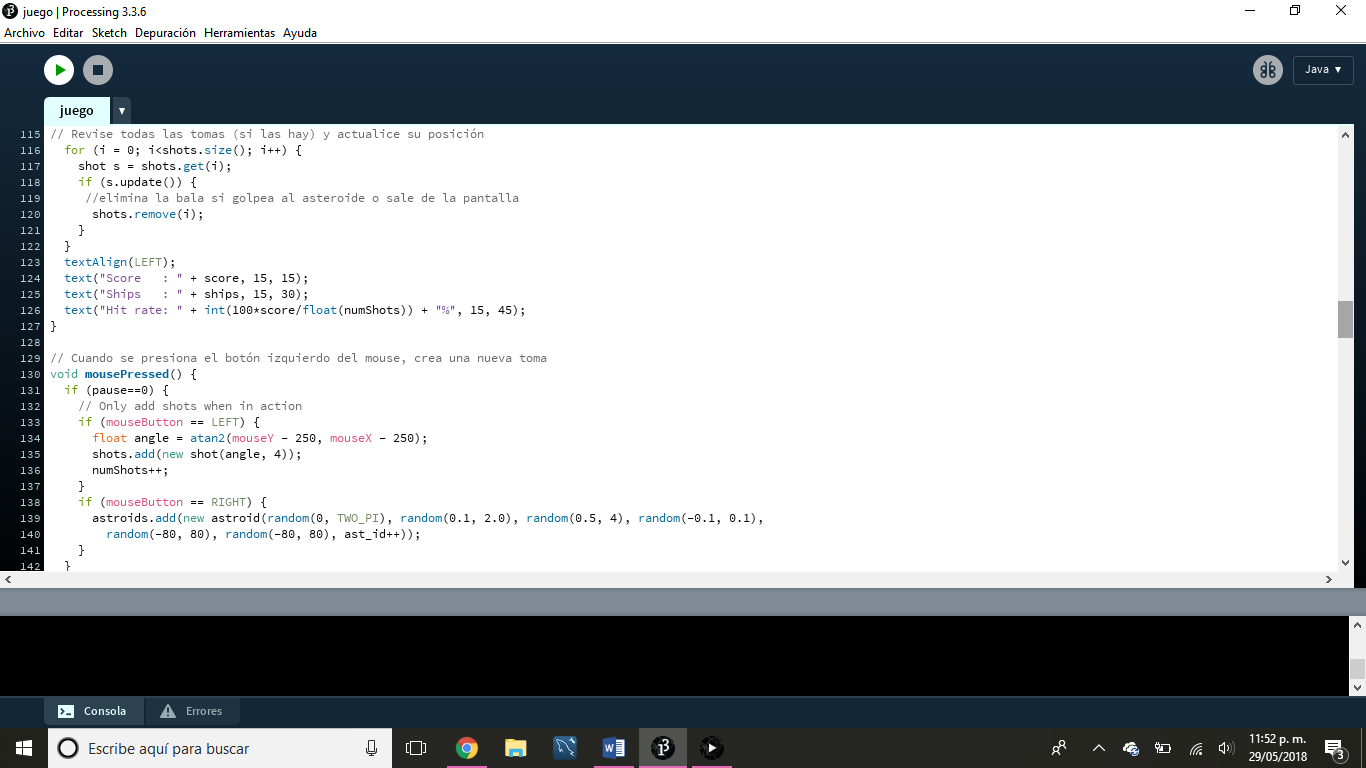
1. Revisamos todas las tomas de balas y que actualice su posición y nos indique el record del juego y vidas.

**Funciones que se utilizaron:**

**Remove:** Remueve un elemento en la lista.

**Get:** Lee el color de cualquier píxel o toma una sección de una imagen. Si no se especifican parámetros, se devuelve la imagen completa.

**size:** Define la dimensión del ancho y alto de la ventana de visualización en unidades de píxeles.

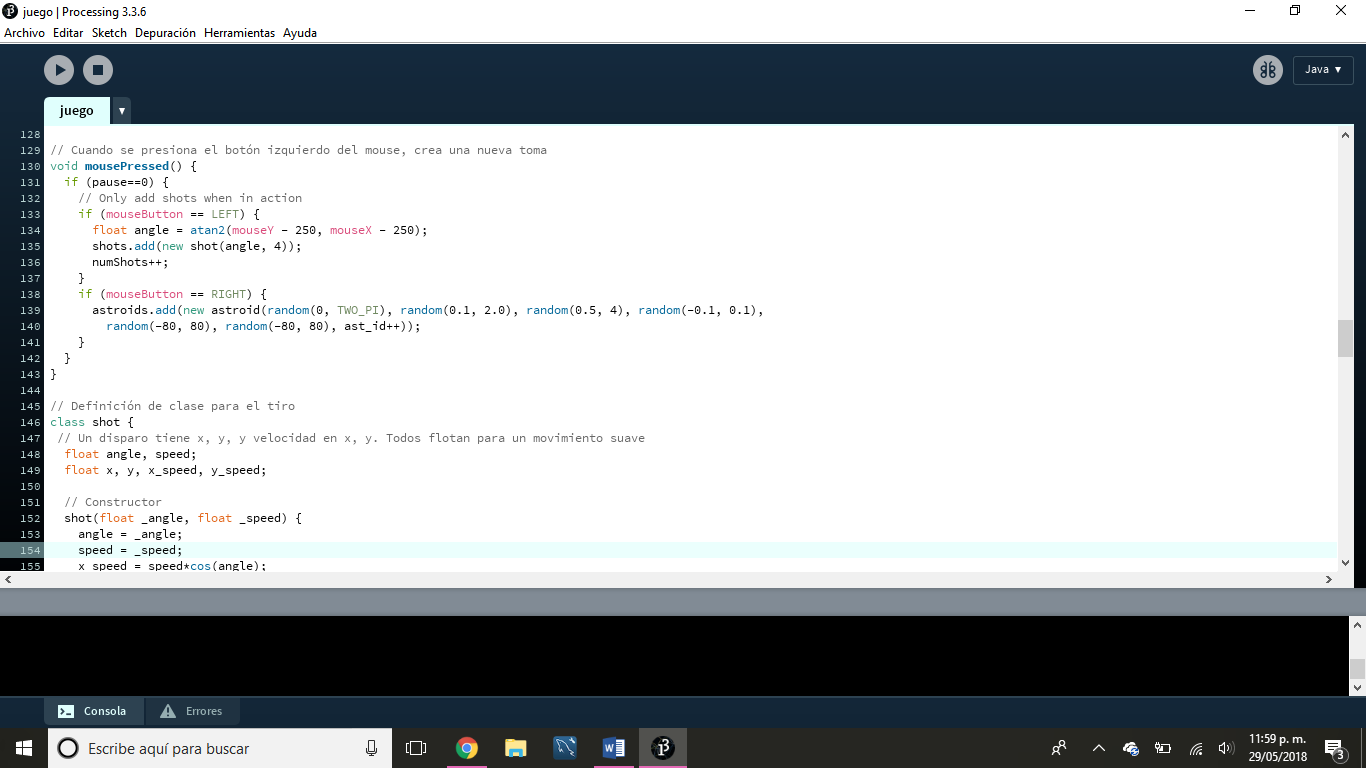


1. Cuando presionamos el botón izquierdo del mouse se realizan disparos.

**Funciones que se utilizaron:**

**Random:** Genera números aleatorios. Cada vez que se llama a la función random (), devuelve un valor inesperado dentro del rango especificado. Si solo se pasa un parámetro a la función, devolverá un flotante entre cero y el valor del parámetro alto.

**Add:** Agrega un nuevo elemento a la lista.

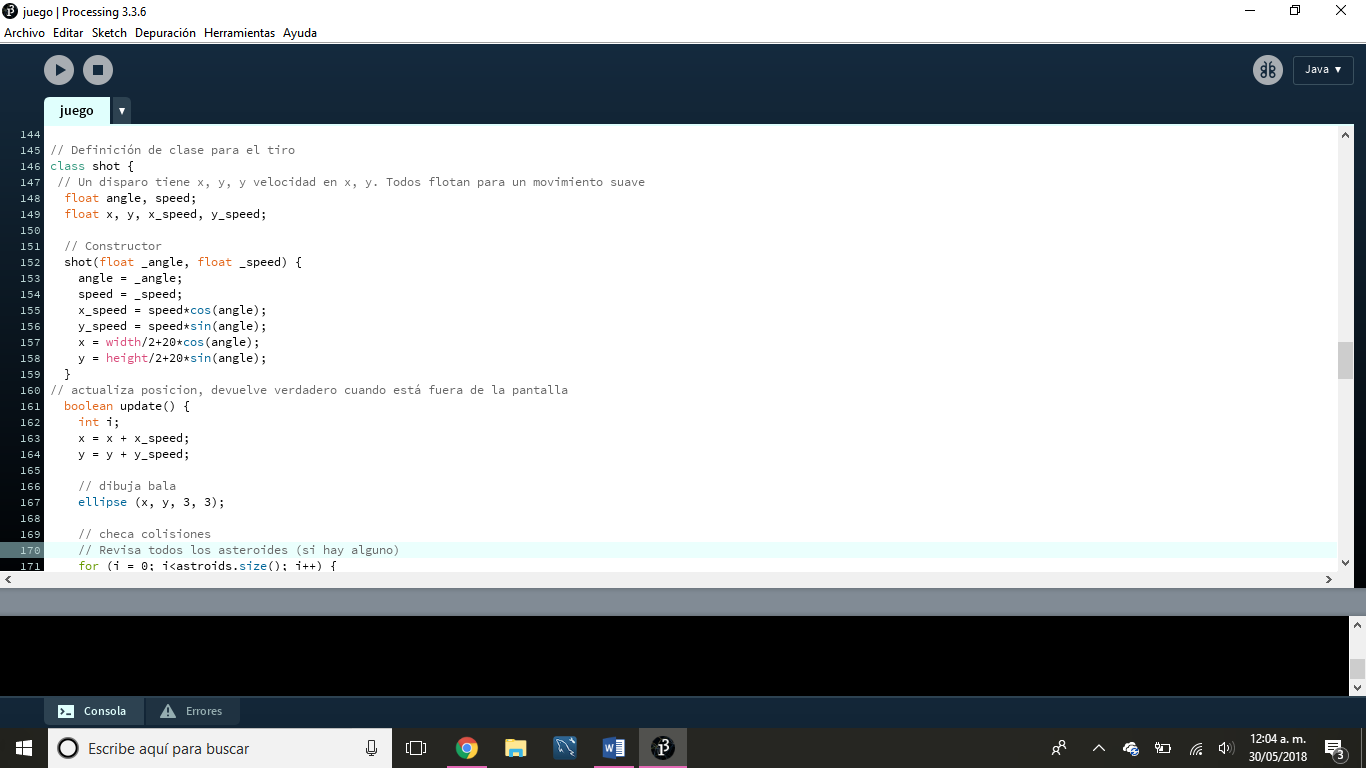


1. Definimos la clase que nos servirá para los tiros.

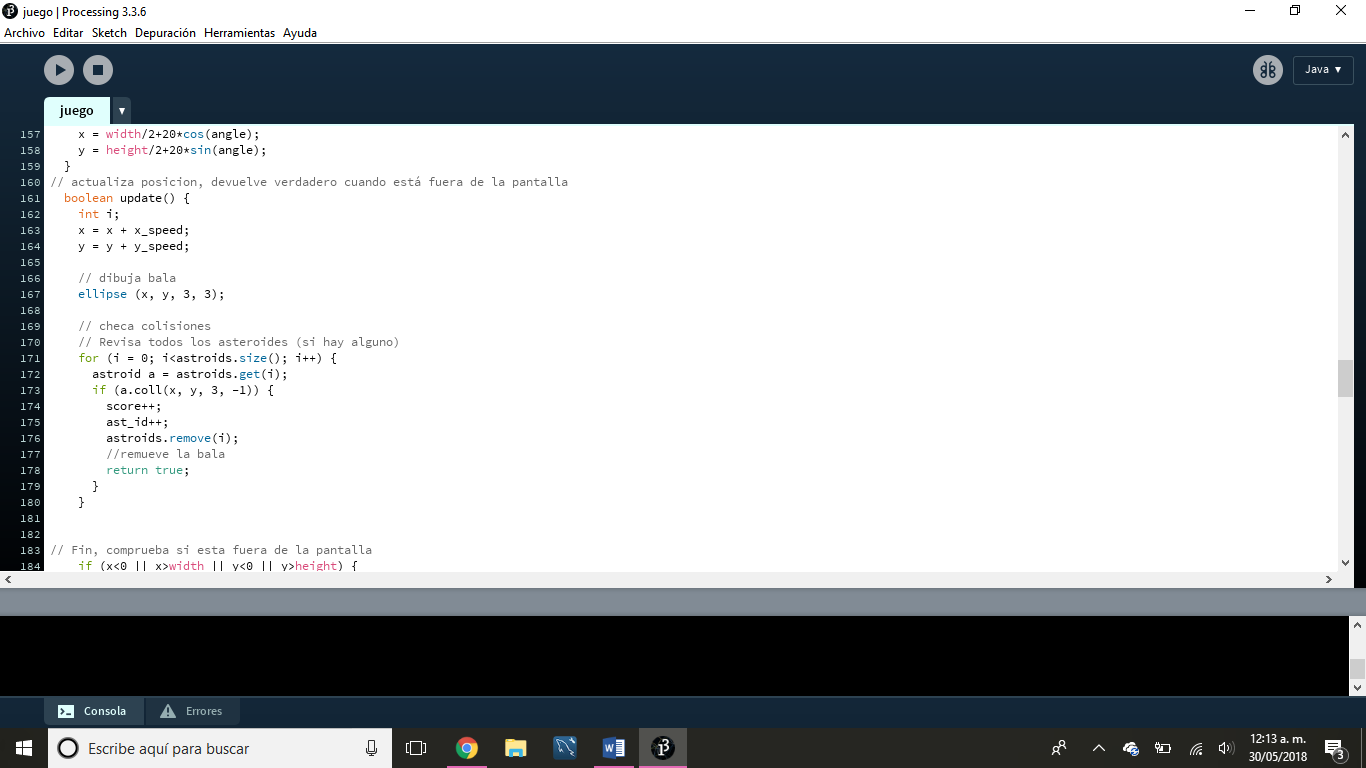
**Funciones que se utilizaron:**

**Class:**

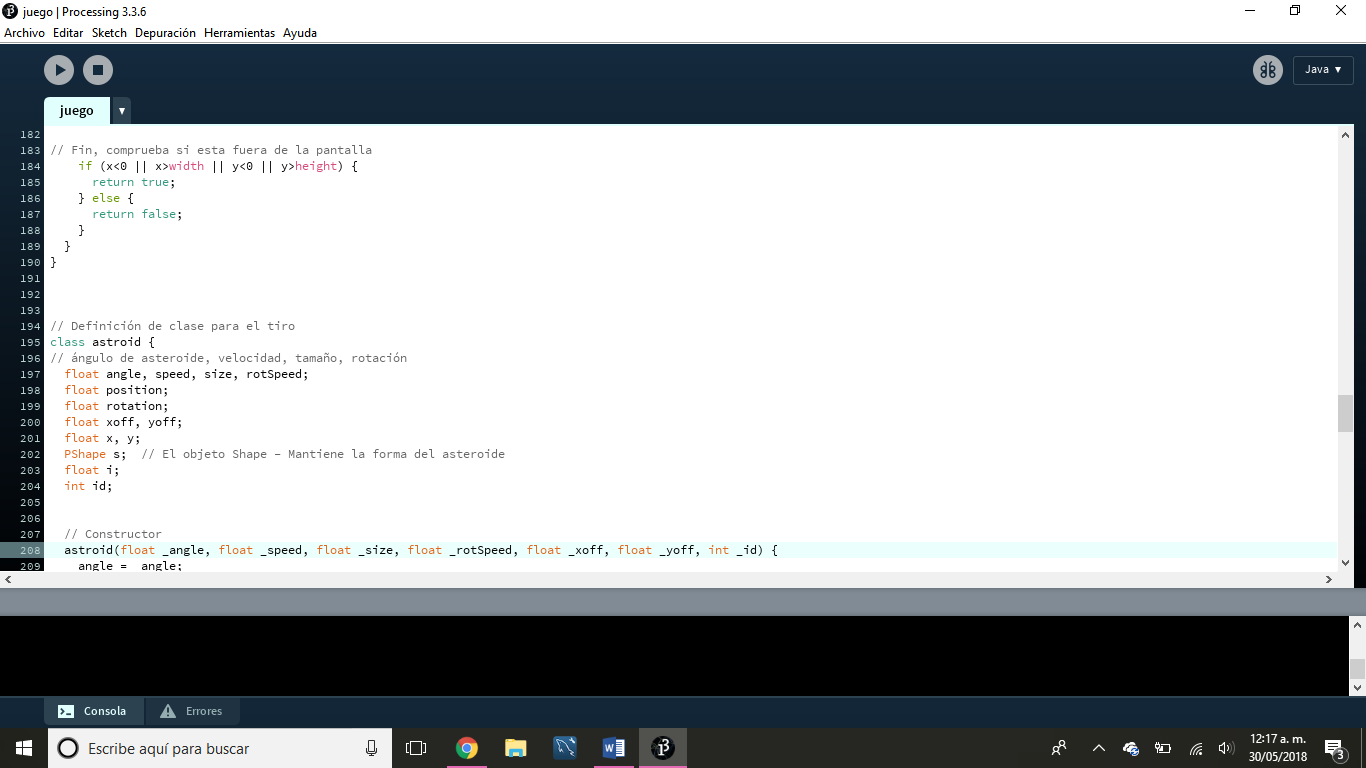
Palabra clave utilizada para indicar la declaración de una clase. Y definimos las variables que utilizaremos para el disparo las cuales las declararemos de tipo float.



1. Actualiza el estado de la bala y nos devuelve true cuando la bala se encuentre fuera de la pantalla.



1. Definimos la clase para el asteroide en la cual definiremos la velocidad , tamaño y rotación con tipo float.



1. Creamos un constructor

**Ellipse:** Función que crea un elipse ejemplo: Ellipse (x,y, width,height).

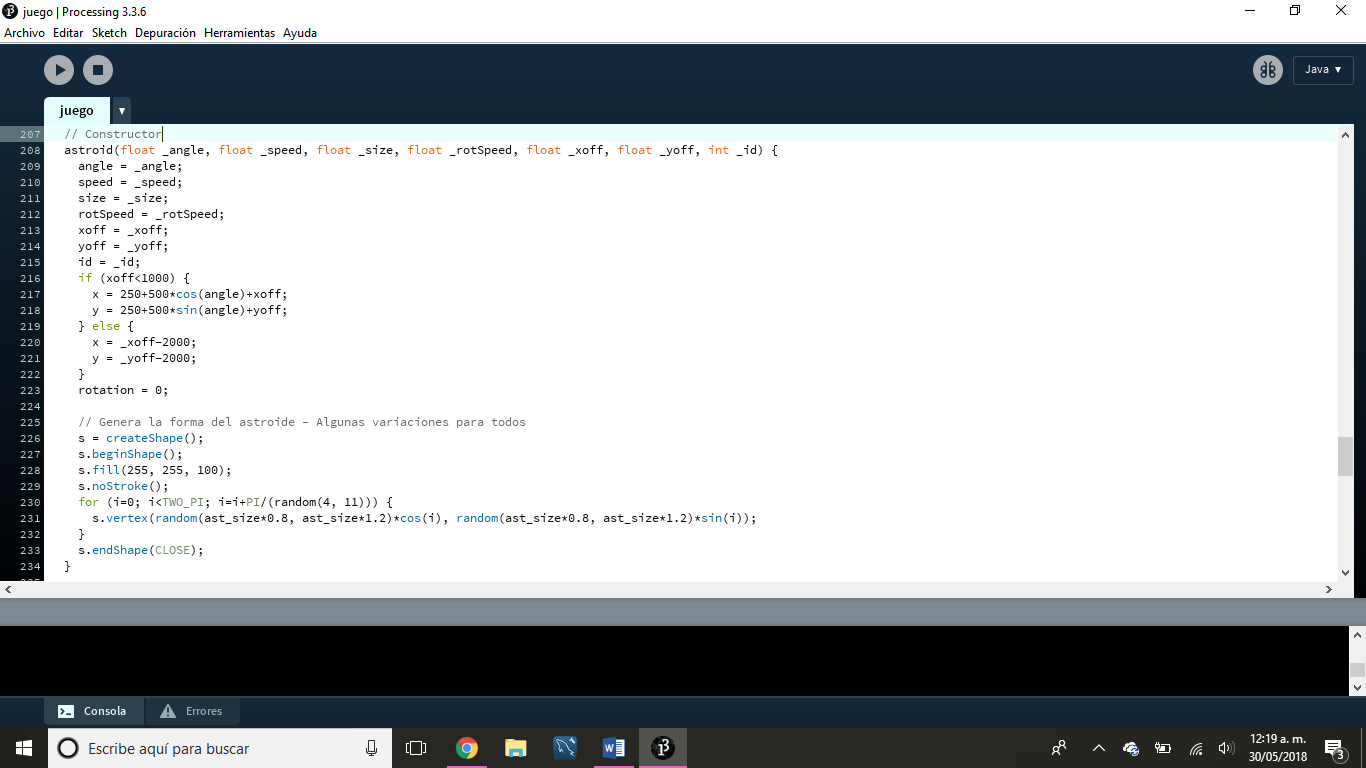
**createShape:** se usa para definir una nueva forma. Una vez creada, esta forma se puede dibujar con la función de forma ()

**beginShape:** comienza a grabar vértices para una forma

**noStroke:** desactiva el dibujo del trazo(contorno). Si se invoca tanto noStroke() como noFill(), no se dibujara nada en la pantalla.

**vertex:** se usa para especificar las coordenadas del vértice para puntos,líneas,triángulos.cuadrantes y polígonos se usa exclusivamente dentro de las funciones beginShape() y endShape().

**endShape**:es la acompañante de beginShape() y solo se puede invocar después de este. Cuando se llama un endShape(), todos los datos de la imagen definidos desde la llamada anterior se escriben en el buffer de imagen.



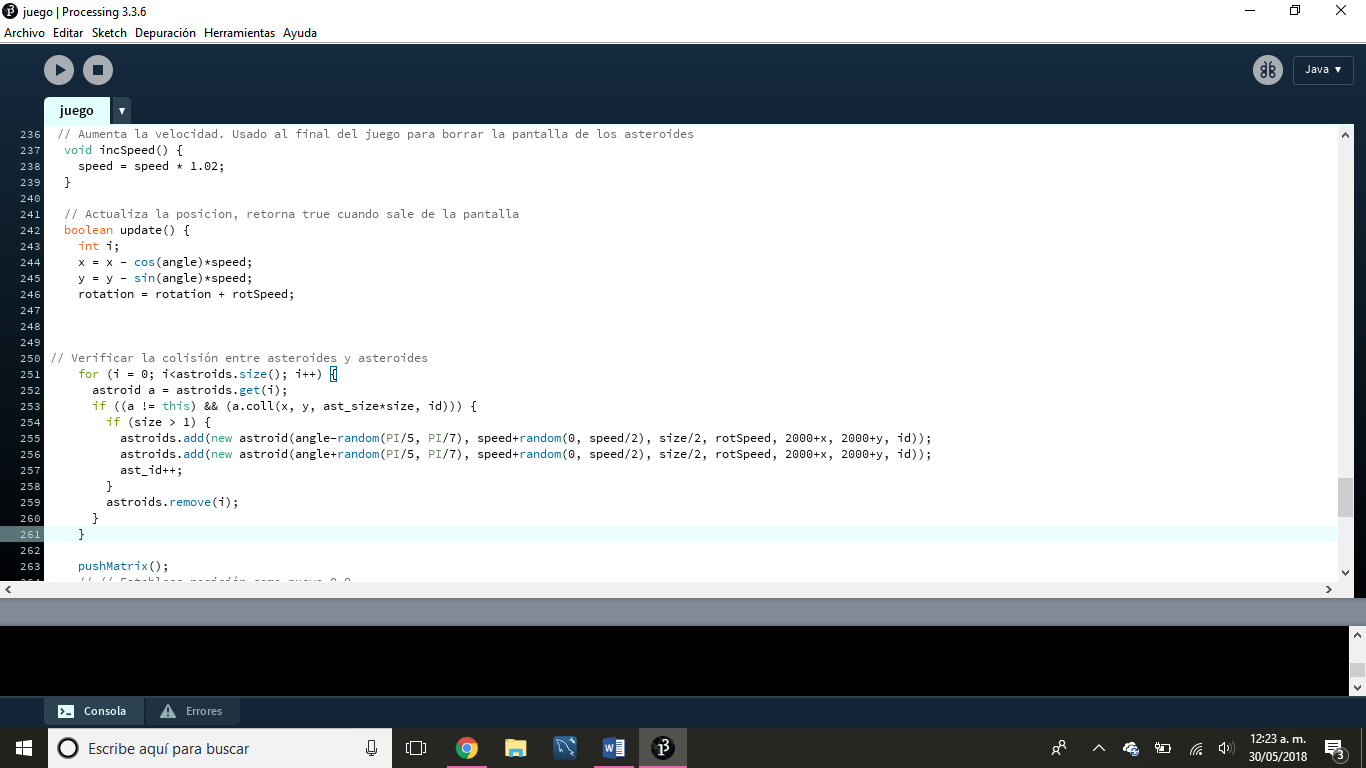
1. Aumentamos la velocidad al final del juego para borrar los asteroides, actualizamos la posición y retornamos el valor de true cuando sale de la pantalla.

**Funciones que se utilizaron:**

**Sin:** Calcula el seno de un ángulo. Esta función espera que los valores del parámetro de ángulo se proporcionen en radianes (valores de 0 a 6.28). Los valores se devuelven en el rango de -1 a 1.

**Cos:** Calcula el coseno de un ángulo. Esta función espera que los valores del parámetro de ángulo se proporcionen en radianes (valores de 0 a PI \* 2). Los valores se devuelven en el rango de -1 a 1.

**Add:** Agrega un elemento a la lista.



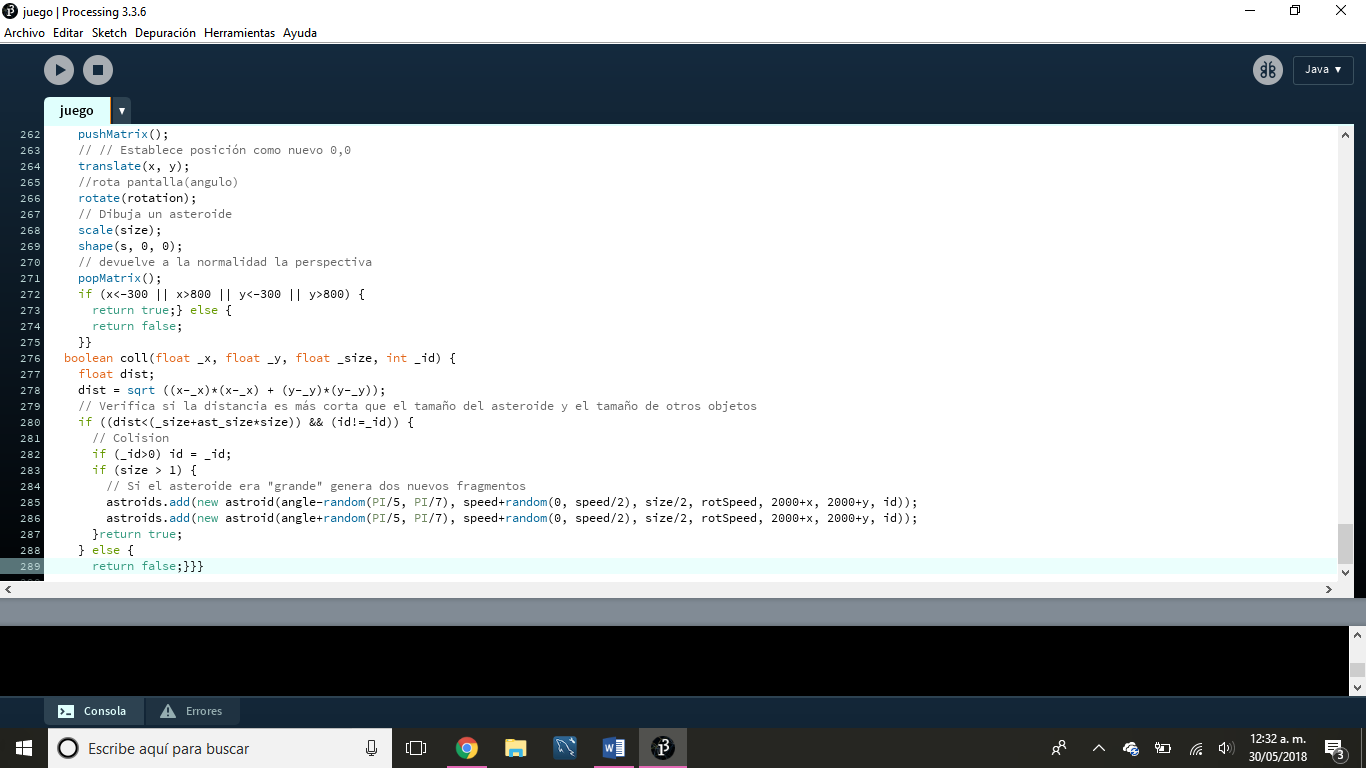
1. Finalmente reiniciamos los procesos y verificamos distancias entre la nave y asteroides

**Funciones que se utilizaron:**

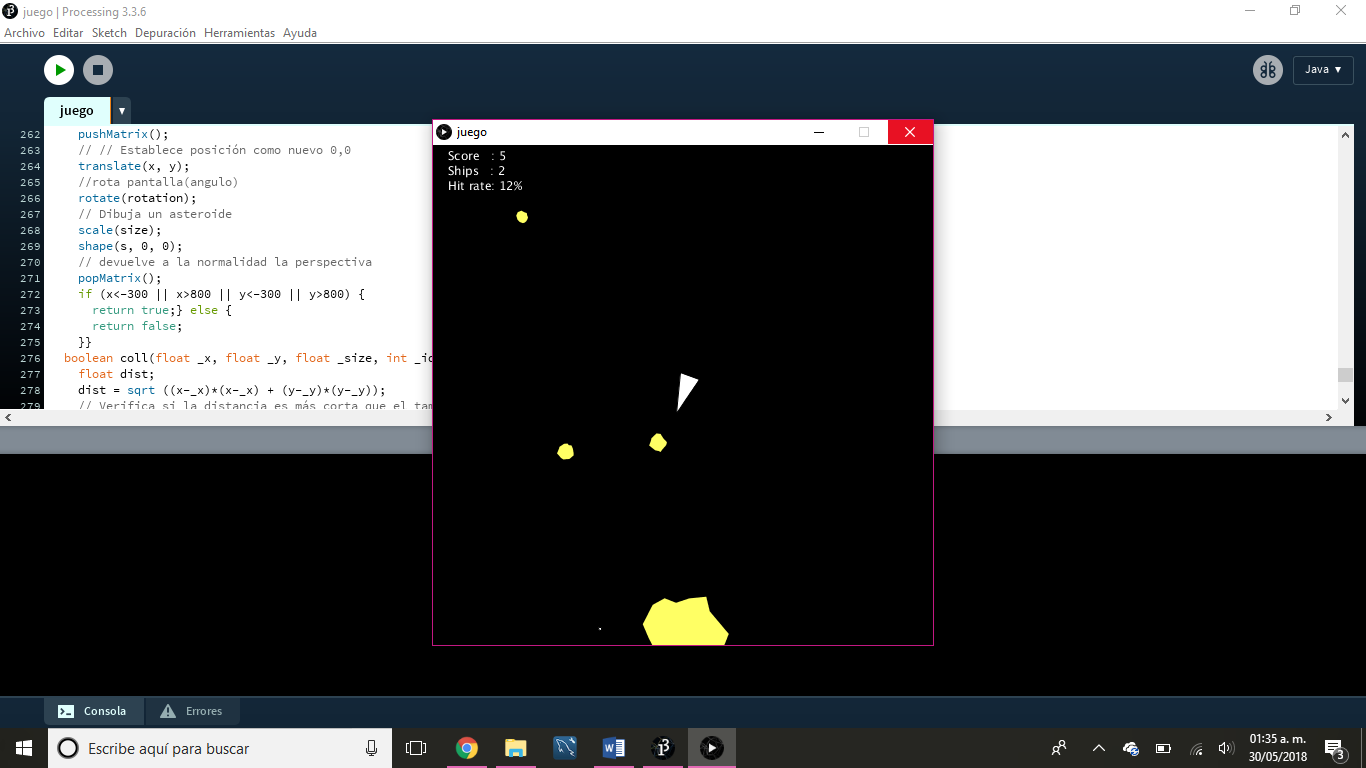
**scale:** aumenta o disminuye el tamaño de una forma al expsandir y contraer vértices.

**shape:** dibuja formas en la ventana de visualización.

**sqrt:** Calcula la raíz cuadrada de un número.



Resultado final:



**Conclusiones**

* Guadalupe García Cárdenas

Llegue a la conclusión de que Procesing cuenta con grandes beneficios y que es muy fácil de ocupar, al igual que cuenta con grandes foros en los cuales podemos obtener información y guiarnos en la elaboración de ciertos programas o proyectos que podamos realizar, sus funciones son muy fáciles de utilizar y no se complica ya que contamos con gran información acerca de Processing.

* Gerardo Sanchez Romero

Mi conclusión es que para utilizar Procesing debemos tener un poco de conocimientos en java para que se nos facilite la programación en este, no obstante no logra ser difícil ya que contamos con mucha información, al igual que varios sitios en internet que nos brindan gran información acerca de varios temas que podamos pretender desarrollar.