 **TECNOLÓGICO NACIONAL DE MÉXICO**

**INSTITUTO TECNOLÓGICO DE IGUALA**

**MATERIA: GRAFICACIÓN**

Documentación del Juego de Asteroides.

PROFESOR: ING. ARTURO CARLOS RODRÍGUEZ ROMÁN

EQUIPO:

**EDOARDO GUZMAN ALBARRAN**

**LEONARDO FLORES DE LOS SANTOS**

INGENIERÍA EN SISTEMAS COMPUTACIONALES

SEMESTRE: 6º GRUPO: “B” AULA: E-3

Iguala de la independencia Gro. a 31 de junio del 2018.

INTRODUCCION

En esta documentación hablaremos sobre cómo elaborar un juego de asteroides con las herramientas que fuimos adquiriendo conforme pasaba el semestre hasta la actualidad del curso.

Codigo.

Ship ship;

boolean upPressed = false;

boolean downPressed = false;

boolean aPressed = false;

boolean dPressed = false;

float shipSpeed = 2;

float rotationAngle = .2;

float bulletSpeed = 10;

int numAsteroids = 1;

int startingRadius = 50;

PImage[] asteroidPics = new PImage[3];

float bgColor = 0;

PImage naves;

ArrayList<Exhaust> exhaust;

ArrayList<Exhaust> fire;

ArrayList<Bullet> bullets;

ArrayList<Asteroid> asteroids;

PFont font;

int darkCounter;

int darkCounterLimit = 24\*2;

int MAX\_LIVES = 3;

int lives;

int stage = -1;

int diffCurve = 2;

void setup(){

background(bgColor);

size(800,500);

font = createFont("Cambria", 32);

asteroidPics[0] = loadImage("Asteroide-1.jpg");

asteroidPics[1] = loadImage("Asteroide-2.jpg");

asteroidPics[2] = loadImage("Asteroide-3.jpg");

naves = loadImage("nave.jpg");

frameRate(24);

lives = 3;

asteroids = new ArrayList<Asteroid>(0);

}

void draw(){

if( lives >= 0 && asteroids.size()>0){

float theta = heading2D(ship.rotation)+PI/2;

background(0);

ship.update(exhaust, fire);

ship.edges();

ship.render();

if(ship.checkCollision(asteroids)){

lives--;

ship = new Ship();

}

if(aPressed){

rotate2D(ship.rotation,-rotationAngle);

}

if(dPressed){

rotate2D(ship.rotation, rotationAngle);

}

if(upPressed){

ship.acceleration = new PVector(0,shipSpeed);

rotate2D(ship.acceleration, theta);

}

for(Exhaust e: exhaust){

e.update();

e.render();

}

for(Exhaust e: fire){

e.update();

e.render();

}

for(int i = 0; i < bullets.size(); i++){

bullets.get(i).edges();

if(bullets.get(i).update()){

bullets.remove(i);

i--;

}

if(i < 0){

break;

}

bullets.get(i).render();

if(bullets.get(i).checkCollision(asteroids)){

bullets.remove(i);

i--;

}

}

while(exhaust.size() > 20){

exhaust.remove(0);

}

while(fire.size()>3){

fire.remove(0);

}

while(bullets.size() > 30){

bullets.remove(0);

}

for(Asteroid a : asteroids){

a.update();

a.edges();

a.render();

}

for(int i = 0; i < lives; i++){

image(naves,40\*i + 10,ship.r\*1.5,2\*ship.r,3\*ship.r);

}

} else if(lives < 0){

if(darkCounter < darkCounterLimit){

background(0);

darkCounter++;

for(Asteroid a : asteroids){

a.update();

a.edges();

a.render();

}

fill(0, 255-(darkCounterLimit-darkCounter)\*3);

rect(0,0,width,height);

} else {

background(0);

for(Asteroid a : asteroids){

a.update();

a.edges();

a.render();

}

image(naves,width/2 - 5 \* ship.r,height/2-7.5\*ship.r,10\*ship.r,15\*ship.r);

textFont(font, 33);

fill(0, 200);

text("HAS PERDIDO ", width/2-80-2, height\*.75-1);

textFont(font, 32);

fill(255);

text("perdiste vuelve a intentar ", width/2-80, height\*.75);

textFont(font, 16);

fill(0, 200);

text("Has click en el mouse para jugar de nuevo", width/2-80-2, height\*.9-1);

textFont(font, 15);

fill(255);

text("Has click en el mouse para jugar de nuevo", width/2-80, height\*.9);

}

} else {

background(0);

ship = new Ship();

ship.render();

textFont(font, 32);

fill(255);

if(stage > -1){

text("combate " + (stage + 1) + " Completo", width/2-120, height/2);

} else {

text("asteroid war ", width/2-80, height/2);

}

textFont(font, 15);

fill(255);

text("Has click en el mousesi si estas preparado " + (stage + 2), width/2-100, height\*.75);

}

}

void mousePressed(){

if(lives < 0){

stage = -1;

lives = 3;

asteroids = new ArrayList<Asteroid>(0);

} else if (asteroids.size()==0){

stage++;

reset();

}

}

void reset(){

ship = new Ship();

exhaust = new ArrayList<Exhaust>();

fire = new ArrayList<Exhaust>();

bullets = new ArrayList<Bullet>();

asteroids = new ArrayList<Asteroid>();

for(int i = 0; i <numAsteroids + diffCurve\*stage; i++){

PVector position = new PVector((int)(Math.random()\*width), (int)(Math.random()\*height-100));

asteroids.add(new Asteroid(position, startingRadius, asteroidPics, stage));

}

darkCounter = 0;

}

void fireBullet(){

PVector pos = new PVector(0, ship.r\*2);

rotate2D(pos,heading2D(ship.rotation) + PI/2);

pos.add(ship.position);

PVector vel = new PVector(0, bulletSpeed);

rotate2D(vel, heading2D(ship.rotation) + PI/2);

bullets.add(new Bullet(pos, vel));

}

void keyPressed(){

if(key==CODED){

if(keyCode==UP){

upPressed=true;

} else if(keyCode==DOWN){

downPressed=true;

} else if(keyCode == LEFT){

aPressed = true;

}else if(keyCode==RIGHT){

dPressed = true;

}

}

if(key == 'a'){

aPressed = true;

}

if(key=='d'){

dPressed = true;

}

if(key=='w'){

upPressed=true;

}

if(key=='s'){

downPressed=true;

}

}

void keyReleased(){

if(key==CODED){

if(keyCode==UP){

upPressed=false;

ship.acceleration = new PVector(0,0);

} else if(keyCode==DOWN){

downPressed=false;

ship.acceleration = new PVector(0,0);

} else if(keyCode==LEFT){

aPressed = false;

} else if(keyCode==RIGHT){

dPressed = false;

}

}

if(key=='a'){

aPressed = false;

}

if(key=='d'){

dPressed = false;

}

if(key=='w'){

upPressed=false;

ship.acceleration = new PVector(0,0);

}

if(key=='s'){

downPressed=false;

ship.acceleration = new PVector(0,0);

}

if(key == ' '){

fireBullet();

}

}

float heading2D(PVector pvect){

return (float)(Math.atan2(pvect.y, pvect.x));

}

void rotate2D(PVector v, float theta) {

float xTemp = v.x;

v.x = v.x\*cos(theta) - v.y\*sin(theta);

v.y = xTemp\*sin(theta) + v.y\*cos(theta);

}

class Asteroid{

float radius;

float omegaLimit = .05;

PVector position;

PVector velocity;

PVector rotation;

float spin;

int col = 100;

PImage pics[];

PImage pic;

int stage;

float dampening = 1;

public Asteroid(PVector pos, float radius\_, PImage[] pics\_, int stage\_){

radius = radius\_;

stage = stage\_;

position = pos;

float angle = random(2 \* PI);

velocity = new PVector(cos(angle), sin(angle));

velocity.mult((50\*50)/(radius\*radius));

velocity.mult(sqrt(stage + 2));

velocity.mult(dampening);

angle = random(2 \* PI);

rotation = new PVector(cos(angle), sin(angle));

spin = (float)(Math.random()\*omegaLimit-omegaLimit/2);

int rnd = (int)(Math.random()\*3);

pics = pics\_;

pic = pics[rnd];

}

void breakUp(ArrayList<Asteroid> asteroids){

if(radius <= 30){

asteroids.remove(this);

} else if (radius < 33){

for(int i = 0; i < 2; i++){

float angle = random(2\*PI);

PVector rand = new PVector(radius\*sin(angle), radius\*cos(angle));

rand.add(position);

asteroids.add(new Asteroid(rand, radius\*.8, pics, stage));

}

asteroids.remove(this);

} else {

for(int i = 0; i < 3; i++){

float angle = random(2\*PI);

PVector rand = new PVector(radius\*sin(angle), radius\*cos(angle));

rand.add(position);

asteroids.add(new Asteroid(rand, radius\*.8, pics, stage));

}

asteroids.remove(this);

}

}

void update(){

position.add(velocity);

rotate2D(rotation, spin);

}

void render(){

fill(col);

circ(position.x, position.y);

if (position.x < radius){

circ(position.x + width, position.y);

} else if (position.x > width-radius) {

circ( position.x-width, position.y);

}

if (position.y < radius) {

circ(position.x, position.y + height);

} else if (position.y > height-radius){

circ(position.x, position.y-height);

}

}

void edges(){

if (position.x < 0){

position.x = width;

}

if (position.y < 0) {

position.y = height;

}

if (position.x > width) {

position.x = 0;

}

if (position.y > height){

position.y = 0;

}

}

void circ(float x, float y){

pushMatrix();

translate(x,y);

rotate(heading2D(rotation)+PI/2);

image(pic, -radius,-radius,radius\*2, radius\*2);

popMatrix();

}

float heading2D(PVector pvect){

return (float)(Math.atan2(pvect.y, pvect.x));

}

void rotate2D(PVector v, float theta) {

float xTemp = v.x;

v.x = v.x\*cos(theta) - v.y\*sin(theta);

v.y = xTemp\*sin(theta) + v.y\*cos(theta);

}

}

class Bullet{

PVector position;

PVector velocity;

int radius = 5;

int counter = 0;

int timeOut = 24 \* 2;

float alpha;

PImage img = loadImage("laser.png");

public Bullet(PVector pos, PVector vel){

position = pos;

velocity = vel;

alpha = 255;

}

void edges(){

if (position.x < 0){

position.x = width;

}

if (position.y < 0) {

position.y = height;

}

if (position.x > width) {

position.x = 0;

}

if (position.y > height){

position.y = 0;

}

}

boolean checkCollision(ArrayList<Asteroid> asteroids){

for(Asteroid a : asteroids){

PVector dist = PVector.sub(position, a.position);

if(dist.mag() < a.radius){

a.breakUp(asteroids);

return true;

}

}

return false;

}

boolean update(){

alpha \*= .9;

counter++;

if(counter>=timeOut){

return true;

}

position.add(velocity);

return false;

}

void render(){

fill(255);

pushMatrix();

translate(position.x, position.y);

rotate(heading2D(velocity)+PI/2);

//ellipse(0,0, radius, radius\*5);

image(img, -radius/2, -2\*radius, radius, radius\*5);

popMatrix();

}

float heading2D(PVector pvect){

return (float)(Math.atan2(pvect.y, pvect.x));

}

}

class Exhaust{

PVector position;

PVector velocity;

float diameter;

color hugh;

public Exhaust(PVector pos, PVector vel, color col, int rad){

position = pos;

velocity = vel;

diameter = (float)(Math.random()\*rad);

hugh = col;

}

void render(){

noStroke();

fill(hugh);

ellipse(position.x, position.y, diameter, diameter);

}

void update(){

position.add(velocity);

velocity.mult(.9);

}

}

class Ship{

PVector position;

PVector velocity;

PVector acceleration;

PVector rotation;

float drag = .9;

float r = 15;

PImage img = loadImage("nave.jpg");

public Ship(){

position = new PVector(width/2, height-50);

acceleration = new PVector(0,0);

velocity = new PVector(0,0);

rotation = new PVector(0,1);

}

void update(ArrayList<Exhaust> exhaust, ArrayList<Exhaust> fire){

PVector below = new PVector(0, -2\*r);

rotate2D(below, heading2D(rotation)+PI/2);

below.add(position);

color grey = color(100, 75);

int exhaustVolume = (int)(velocity.mag())+1;

for(int i = 0; i <exhaustVolume; i++){

float angle = (float)(Math.random()\*.5-.25);

angle += heading2D(rotation);

PVector outDir = new PVector(cos(angle), sin(angle));

exhaust.add(new Exhaust(below, outDir, grey, 15));

}

for(int i = 0; i <1; i++){

float angle = (float)(Math.random()\*.5-.25);

angle += heading2D(rotation);

PVector outDir = new PVector(cos(angle), sin(angle));

outDir.y = 0;

below.add(outDir);

below.y-=.5;

color red = color((int)(200 + Math.random()\*55),(int)( 150+Math.random()\*105), 50, 250);

fire.add(new Exhaust(below,outDir, red, 5));

}

velocity.add(acceleration);

velocity.mult(drag);

velocity.limit(5);

position.add(velocity);

}

void edges(){

if (position.x < r){

position.x = width-r;

}

if (position.y < r) {

position.y = height-r;

}

if (position.x > width-r) {

position.x = r;

}

if (position.y > height-r){

position.y = r;

}

}

boolean checkCollision(ArrayList<Asteroid> asteroids){

for(Asteroid a : asteroids){

PVector dist = PVector.sub(a.position, position);

if(dist.mag() < a.radius + r/2){

a.breakUp(asteroids);

return true;

}

}

return false;

}

void render(){

float theta = heading2D(rotation) + PI/2;

theta += PI;

pushMatrix();

translate(position.x, position.y);

rotate(theta);

fill(0);

image(img,-r,-r\*1.5,2\*r,3\*r);

popMatrix();

}

float heading2D(PVector pvect){

return (float)(Math.atan2(pvect.y, pvect.x));

}

void rotate2D(PVector v, float theta) {

float xTemp = v.x;

v.x = v.x\*cos(theta) - v.y\*sin(theta);

v.y = xTemp\*sin(theta) + v.y\*cos(theta);

}

}

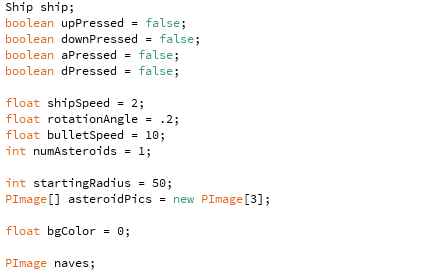


En la materia de graficacion aprendimos varios métodos y sentencias para poder realizar el proyecto.

FUNCIONES

* Size():
* Background()
* Fill
* Height y Width
* Rec()
* PImage()
* TextFont ()
* Text ()
* mousePressed ()
* keyPressed ()
* keyReleased ()
* rotate2D ()
* heading2D ()
* random ()
* sqrt ()
* pushMatrix ()
* translate ()
* popMatrix ()
* color ()
* ellipse ()

En graficacion vimos metodos y variables. Como se muestra en el programa.



En esta seccion se declaran las variables.



El background () se utiliza para cambiar el color del fondo

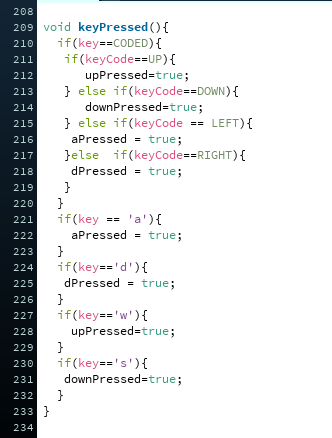
Size () para poner el tamaño del lienzo.

Para que juego se pueda mover y aventar el lasér hay que puchar la tecla espaciadora, en esta se ocupan las funciones

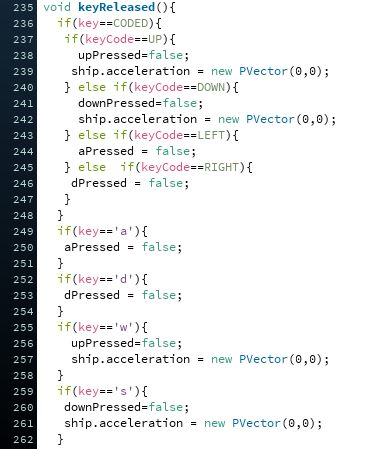
void mousePressed ()



keyPressed. Usaremos la función keyPressed en nuestro para que cada que movamos nuestro coete con las flechas el coete se moverá a donde le hayamos indicado ya se arriba, abajo, izquierda, derecha.



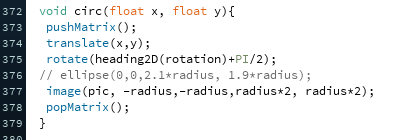
keyReleased funciona cada vez que un botón o tecla es liberada entrará en función.



pushMatrix (). - Impulsa la matriz de transformación actual a la pila de la matriz. La función pushMatrix () guarda el sistema de coordenadas actual en la pila y popMatrix () restaura el sistema de coordenadas anterior. pushMatrix () y popMatrix () se usan en conjunción con las otras funciones de transformación y se pueden incorporar para controlar el alcance de las transformaciones.

popMatrix (). - Coloca la matriz de transformación actual fuera de la pila de la matriz.

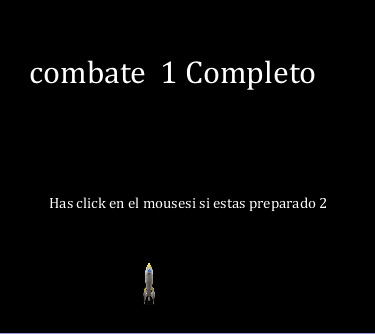
Translate (). - Especifica una cantidad para desplazar objetos dentro de la ventana de visualización. El parámetro x especifica la traducción izquierda / derecha, el parámetro y especifica la conversión ascendente / descendente, y el parámetro z especifica las traducciones hacia / desde la pantalla.

En estas 3 funciones estamos especificando los asteroides, como van a girar y hacia dónde van a estar girando.

Ellipse (). - El ellipse es usando como un ovalo pero en nuestro código del juego lo usamos para darle diseño a los asteroides.

Color (). - En color le asignamos el color que nosotros queramos usar, en este caso nosotros usamos el color gris a los asteroides.







CONCLUSION

El programa Processing es muy amigable y sencillo de usar para el usuario ya que trae herramientas (librerías) que con solo ponerlas hace lo que quieres que haga

Al imitar el juego de asteroides fue un gran reto ya que teníamos que implementar todo lo visto y pues el resultado es el siguiente.

La graficación nos permite a ser imágenes gráficas, juegos en 3D mediante la computadora. Existen varias áreas en las que la graficación puede ser usada como una forma para mejorar la representación visual de diversos recursos la graficacion en nuestra carrera de ing. En sistemas computacionales nos a servido de mucha ayuda ya que hemos aprendidos nuevos conocimientos y estos los podemos aplicar en otras áreas como Puede en la medicina, educación, artes, diseño, entre otras, a pesar de que estas especialidades no tienen que ver una con otra. Es por esto que la graficación es muy importante, ya que facilita la forma en que se ven diversas cosas, por ejemplo, puede ayudar a crear códigos para juegos o aplicaciones que deseen.