Abre la clase Menu.as y agrega las líneas que se muestran a continuación:

```
//Menu.as
package states
{
   import com.greensock.TimelineMax;
   import com.greensock.TweenLite;
   import core.Assets;
   import core.Game;
   import flash.media.SoundChannel;
   import interfaces.IState;
   import objects.Background;
   import starling.display.Button;
   import starling.display.Image;
   import starling.display.Sprite;
   import starling.events.EnterFrameEvent;
   import starling.events.Event;

   /**
    * ...
    * @author ...
    */
   public class Menu extends Sprite implements IState
   {
        private var game:Game;
        private var but_play:Button;
        private var menu_bg:Background;
        private var game_logo:Image;
        private var art:Image;
        private var myChannel:SoundChannel = new
SoundChannel();

        public function Menu(game:Game)
        {
               this.game = game;
               addEventListener(Event.ADDED_TO_STAGE, init);

        }

        private function init(e:Event):void
        {
               removeEventListener(Event.ADDED_TO_STAGE, init);

               //add the background
               menu_bg = new Background();
               //selecciona la textura, si así lo deseas
               //menu_bg.bg_texture = Assets.bg_2;
               addChild(menu_bg);

               //add the buttons
               but_play = new
Button(Assets.ta.getTexture("but_play.png"), "");
               addChild(but_play);
               but_play.x = stage.stageWidth / 2;
```

```
            but_play.y = 450;
            //pivote al centro
            but_play.pivotX = but_play.width * 0.5;
            but_play.pivotY = but_play.height * 0.5;
            //hagamoslo más pequeño
            but_play.scaleX = 0.5;
            but_play.scaleY = 0.5;
            but_play.addEventListener(Event.TRIGGERED,
playMe);

            //add the logo
            game_logo = new
Image(Assets.ta.getTexture("game_name.png"));
            //más pequeño
            game_logo.scaleX = 0.5;
            game_logo.scaleY = 0.5;
            //init pos
            game_logo.x = stage.stageWidth / 2 -
game_logo.width / 2;
            game_logo.y = stage.stageHeight;

            addChild(game_logo);

            //agregando el arte
            art = new
Image(Assets.ta.getTexture("fenrir_wolf_ol.png"));
            art.x = stage.stageWidth / 2 - art.width / 2;
            art.y = stage.stageHeight;
            addChild(art);

            //agregando el tween
            TweenLite.to(game_logo, 2, { y:stage.stageHeight
/ 2 - game_logo.height / 2 , delay:2 } );
            TweenLite.to(art, 2, { y:20 , delay:4 } );

            var timeline:TimelineMax = new TimelineMax( {
repeat:-1, yoyo:true, repeatDelay:0 } );
            timeline.to(but_play, 1, { scaleX:0.7,
scaleY:0.7 } );

            //música
            myChannel = Assets.intro_music.play(0, 9999);
      }

      private function playMe(e:Event):void
      {
            game.changeState(Game.PLAY_STATE);
      }

      /* INTERFACE interfaces.IState */

      public function update():void
      {
            //el bg puede moverse...si así lo deseas
```

```
                //menu_bg.update();

                /*var myDate:Date = new Date();
                art.y = 100 +  Math.cos(myDate.getTime() *
0.002) * 25;*/
        }

        public function destroy():void
        {
                //quitamos del display list y lo enviamos al
garbage collector
                removeFromParent(true);
                //quitamos la música
                //música!
                myChannel.stop();
        }

    }

}
```

Observa que en la función que limpia la escena se está deteniendo el canal de música, de lo contrario queda activa en la pantalla de juego. Haz lo mismo con la clase Play.as, agregando la música y el efecto sonoro del choque contra los pájaros:

```
//Play.as
package states
{
    import com.greensock.TweenLite;
    import com.greensock.TweenMax;
    import core.Assets;
    import core.Game;
    import flash.geom.Rectangle;
    import flash.media.SoundChannel;
    import flash.utils.getTimer;
    import interfaces.IState;
    import managers.CollisionManager;
    import managers.FruitsManager;
    import managers.ParticleManager;
    import objects.BgParallax;
    import objects.Dog;
    import objects.BadBirds;
    import starling.core.Starling;
    import starling.display.Image;
    import starling.display.MovieClip;
    import starling.display.Sprite;
    import starling.events.EnterFrameEvent;
    import starling.events.Event;
    import starling.events.Touch;
    import starling.events.TouchEvent;
    import starling.events.TouchPhase;
```

```
   /**
    * ...
    * @author ...
    */
   public class Play extends Sprite implements IState
   {
        public var husky:Dog;
        private var bg:BgParallax;
        public var game:Game;

        //game timer
        private var gameTimer:uint = getTimer();
        private var elapsedTime:uint;


        //bad birds
        private var badBird1:BadBirds = new
BadBirds(Assets.ta.getTextures("redBird_"), 12);
        private var badBird2:BadBirds = new
BadBirds(Assets.ta.getTextures("yellowBird_"), 12);
        private var badBird3:BadBirds = new
BadBirds(Assets.ta.getTextures("greyBird_"), 12);
        private var badBird4:BadBirds = new
BadBirds(Assets.ta.getTextures("blueBird_"), 12);
        private var myAlert:Image;
        private var explosion:MovieClip;
        private var collisionManager:CollisionManager;
        public var fruitManager:FruitsManager;
        public var particleManager:ParticleManager;

        private var myChannel:SoundChannel = new
SoundChannel();

        //public var gameArea:Rectangle;

        public function Play(game:Game)
        {
              this.game = game;

              addEventListener(Event.ADDED_TO_STAGE, init);

        }

        private function init(e:Event):void
        {
              removeEventListener(Event.ADDED_TO_STAGE, init);
              trace("PLAY!");

              //para crear un área de juego por medio de un
rectángulo
              //gameArea = new Rectangle(0, 100,
stage.stageWidth, stage.stageHeight - 250);

              //música
```

```
            myChannel = Assets.play_music.play(0, 9999);

            //inicializamos el timer
            gameTimer = 0;
            elapsedTime = 0;

            addEventListener(EnterFrameEvent.ENTER_FRAME,
checkTimer);

            //creamos el bg
            bg = new BgParallax();
            //bg.speed = 10;
            addChild(bg);

            //creamos al husky
            husky = new
Dog(Assets.ta.getTextures("husky_run_"), 12);
            husky.x = 400;
            husky.y = 400;
            //volteamos al husky hacia la derecha
            husky.scaleX = -1;
            Starling.juggler.add(husky);
            addChild(husky);

            //add enemies
            //recuerda que debes de pasar referencia a esta
pantalla de juego con this
            createEnemies();

            //add fruits!
            fruitManager = new FruitsManager(this);
            //collision manager
            collisionManager = new CollisionManager(this);
            //particles manager
            particleManager = new ParticleManager(this);

            //add alert sign
            myAlert = new
Image(Assets.ta.getTexture("warning-01.png"));
            //colocamos el pivote al centro para una mejor
transformación
            myAlert.pivotX = myAlert.width * 0.5;
            myAlert.pivotY  = myAlert.height * 0.5;
            //escalamos el objeto a 0 para desaparecerlo
momentáneamente
            myAlert.scaleX = myAlert.scaleY = 0;
            addChild(myAlert);
            //posicionamos el warning 45 pixeles de la
orilla derecha
            myAlert.x = stage.stageWidth - 45;

            //creamos la explosión
            explosion = new
MovieClip(Assets.ta.getTextures("explosion_"), 12);
```

```
            explosion.pivotX = explosion.width * 0.5;
            explosion.pivotY  = explosion.height * 0.5;
            addChild(explosion);
            explosion.visible = false;
            explosion.loop = false;

            addEventListener(TouchEvent.TOUCH, dogMove);
        }

        private function checkTimer(e:EnterFrameEvent):void
        {
            //empezamos a calcular el tiempo que ha
transcurrido
            elapsedTime = gameTimer++;
            //evaluamos si pasó el tiempo que queremos para
activar los enemigos
            if (elapsedTime >= 200) {
                gameTimer = 0;
                //esta variable nos permite sacar un
número aleatorio del 0 al 400
                var aBird:uint = Math.ceil(Math.random() *
400);
                trace ("aBird:" + aBird);
                //lo pasamos como parámetro para solicitar
algún tipo de pájaro entre los 4 disponibles
                activateBirds(aBird);
            }
            //test trace
            //trace("duration: " + Math.ceil(gameTimer));
        }

        private function activateBirds(aBird:uint):void
        {
            //evaluamos el valor de aBird para activar los
pájaros requeridos
            //modificamos el valor de attackSpeed para que
la velocidad de los pájaros sea distinta
            if (aBird <= 100) {
                badBird1.y = Math.ceil(Math.random() *
550);
                myAlert.y = badBird1.y;
                TweenLite.to(myAlert, 0.5, {scaleX:0.3,
scaleY:0.3, onComplete:hideAlert } );
                badBird1.attackSpeed = 1.5;
                badBird1.attack();
            } else if (aBird >= 101 && aBird <= 200) { //si
esta entre 101 y 200
                badBird2.y = Math.ceil(Math.random() *
550);
                myAlert.y = badBird2.y;
                TweenLite.to(myAlert, 0.5, {scaleX:0.3,
scaleY:0.3, onComplete:hideAlert } );
                badBird2.attackSpeed = 1.8;
                badBird2.attack();
```

```
			} else if (aBird >= 201 && aBird <= 300) {
				badBird3.y = Math.ceil(Math.random() *
550);
				myAlert.y = badBird3.y;
				TweenLite.to(myAlert, 0.5, {scaleX:0.3,
scaleY:0.3, onComplete:hideAlert } );
				badBird3.attackSpeed = 1.7;
				badBird3.attack();
			} else if (aBird >= 301 && aBird <= 400) {
				badBird4.y = Math.ceil(Math.random() *
550);
				myAlert.y = badBird4.y;
				TweenLite.to(myAlert, 0.5, {scaleX:0.3,
scaleY:0.3, onComplete:hideAlert } );
				badBird4.attackSpeed = 2;
				badBird4.attackWave();
			}

			addEventListener(EnterFrameEvent.ENTER_FRAME,
checkCollisions);
		}

		private function checkCollisions(e:Event):void
		{
			if(badBird1.bounds.intersects(husky.bounds))
			{
				//affect screeen!
				TweenLite.to(this, 0.5, { scaleX:1.2,
scaleY:1.2, onComplete:stabilizeScreen } );

				explosion.x = badBird1.x;
				explosion.y = badBird1.y;
				TweenLite.killTweensOf(badBird1);
				badBird1.reuseBirds();
				explosion.visible = true;
				explosion.currentFrame = 0;
				Starling.juggler.add(explosion);
				Assets.crash_sound.play();
			} else
if(badBird2.bounds.intersects(husky.bounds))
			{
				TweenLite.to(this, 0.5, { scaleX:1.2,
scaleY:1.2, onComplete:stabilizeScreen } );
				explosion.x = badBird2.x;
				explosion.y = badBird2.y;
				TweenLite.killTweensOf(badBird2);
				badBird2.reuseBirds();
				explosion.visible = true;
				explosion.currentFrame = 0;
				Starling.juggler.add(explosion);
				Assets.crash_sound.play();
			} else
if(badBird3.bounds.intersects(husky.bounds))
			{
```

```
                    TweenLite.to(this, 0.5, { scaleX:1.2,
scaleY:1.2, onComplete:stabilizeScreen } );
                    explosion.x = badBird3.x;
                    explosion.y = badBird3.y;
                    TweenLite.killTweensOf(badBird3);
                    badBird3.reuseBirds();
                    explosion.visible = true;
                    explosion.currentFrame = 0;
                    Starling.juggler.add(explosion);
                    Assets.crash_sound.play();
             } else
if(badBird4.bounds.intersects(husky.bounds))
             {
                    TweenLite.to(this, 0.5, { scaleX:1.2,
scaleY:1.2, onComplete:stabilizeScreen } );
                    explosion.x = badBird4.x;
                    explosion.y = badBird4.y;
                    TweenMax.killTweensOf(badBird4);
                    badBird4.reuseBirds();
                    explosion.visible = true;
                    explosion.currentFrame = 0;
                    Starling.juggler.add(explosion);
                    Assets.crash_sound.play();
             }

             explosion.addEventListener(Event.COMPLETE,
explosionComplete);
       }

       private function stabilizeScreen():void
       {
             TweenLite.to(this, 0.5, { scaleX:1, scaleY:1 }
);

       }

       private function explosionComplete(e:Event):void
       {
             explosion.removeEventListener(Event.COMPLETE,
explosionComplete);
             explosion.visible  = false;
             Starling.juggler.remove(explosion);

       }

       //escondemos la alerta escalándola a 0
       private function hideAlert():void
       {
             myAlert.scaleX = myAlert.scaleY = 0;
       }

       //create enemies
       private function createEnemies():void
       {
```

```
                //solamente los agregamos al display list y los
activamos con el juggler
                Starling.juggler.add(badBird1);
                addChild(badBird1);
                trace("YEY");

                Starling.juggler.add(badBird2);
                addChild(badBird2);
                trace("YEY");

                Starling.juggler.add(badBird3);
                addChild(badBird3);
                trace("YEY");

                Starling.juggler.add(badBird4);
                addChild(badBird4);
                trace("YEY");
        }

        private function dogMove(e:TouchEvent):void
        {
                var touch:Touch = e.getTouch(this,
TouchPhase.HOVER);
                if (touch)
                {
                        TweenLite.to(husky, 1, { x:touch.globalX -
20,  y:touch.globalY - 20 } );
                }
        }

        /* INTERFACE interfaces.IState */

        public function update():void
        {
                bg.update();
                fruitManager.update();
                collisionManager.update();
        }


        public function destroy():void
        {
                removeFromParent(true);
                //detenemos la música
                myChannel.stop();
        }

    }

}
```

Pudiste ver cómo se agregó la música y los efectos del choque en la interacción de cada pájaro. Al final se detendrá la música cuando la pantalla de Play sea eliminada. Ahora, observa cómo se coloca el efecto de sonido cuando se captura un ítem; abre la clase CollisionManager.as para agregar lo indicado:

```
//CollisionManager.as
package managers
{
   import core.Assets;
   import core.Game;
   import flash.geom.Point;
   import objects.Fruits;
   import objects.Dog;
   import states.Play;
   /**
    * ...
    * @author ...
    */
   public class CollisionManager
   {
        private var play:Play;
        //variables para detectar distancias entre puntos
        private var p1:Point = new Point();
        private var p2:Point = new Point();

        //variables para el radio de los objetos
        //juega con estos parámetros para ajustar el tamaño del
bounding box de los objetos

        //área sensible de las frutas
        private var fruit_radius:int = 32;
        //área sensible del husky
        private var dog_radius:int= 24;

        //agregamos la referencia a nuestro estado de juego
Play
        public function CollisionManager(play:Play)
        {
              //guardamos la referencia en una variable
              this.play = play;
        }

        //constantemente revisamos si hay colisiones
        public function update():void {
              //metodo que calcula si hay colision entre la
fruta y el husky
              fruitCollected();
        }

        //usando deteccion de rectangulos
        private function fruitCollected():void {
              //accedemos al manager de frutas
              var fm:Array = play.fruitManager.fruits_array;
              var f:Fruits;
```

```
            //revisamos las frutas generadas desde arriba de
la lista del array
            for (var i:int = fm.length - 1; i >= 0; i--) {
                  f = fm[i];

                        if
(fm[i].bounds.intersects(play.husky.bounds)) {
                              Assets.item_sound.play();
                              //llamamos al estado de juego
GAME_OVER de la clase Game

//play.game.changeState(Game.GAME_OVER_STATE);

play.particleManager.spawn(f.x, f.y);

play.fruitManager.destroyFruit(f);
                        }

            }
      }

      //usando detección de radios
      /*private function fruitCollected():void
      {
            //accedemos al manager de frutas
            var fm:Array = play.fruitManager.fruits_array;
            var f:Fruits;
            //revisamos las frutas generadas desde arriba de
la lista del array
            for (var i:int = fm.length - 1; i >= 0; i--) {
                  f = fm[i];
                  //usamos los puntos para ver la distancia
entre puntos
                        //primero vemos la posición del
perro en el espacio
                        p1.x = play.husky.x;
                        p1.y = play.husky.y;
                        //ahora de cada fruta
                        p2.x = f.x;
                        p2.y = f.y;
                  //validamos la distancia entre puntos
                        if (Point.distance(p1, p2) <
fruit_radius + dog_radius) {
                              Assets.item_sound.play();
                              //llamamos al estado de juego
GAME_OVER de la clase Game

//play.game.changeState(Game.GAME_OVER_STATE);

play.particleManager.spawn(f.x, f.y);

play.fruitManager.destroyFruits(f);
                        }
```

```
                }
        }*/

    }

}
```