INNEHÅLLSFÖRTECKNING

EXTRAÖVNING I AV 7 – FOR-LOOPS

Att kunna använda for-loops är väldigt användbart och något som en programmerare måste kunna. Det är inte relevant för de övningar vi gjort, och eventuellt kommer ni inte behöva det till ert projekt, men vill ni ändå träna på det finns en liten mindre övning för det här.

> *Notera att det är ganska lätt att råka skriva oändliga loopar när du testar for-loops. Efter 15 sekunder stänger Flash automatiskt ner din testfil, så om du skriver en oändlig loop är det bara att tålmodigt vänta tills programmet svarar igen.*

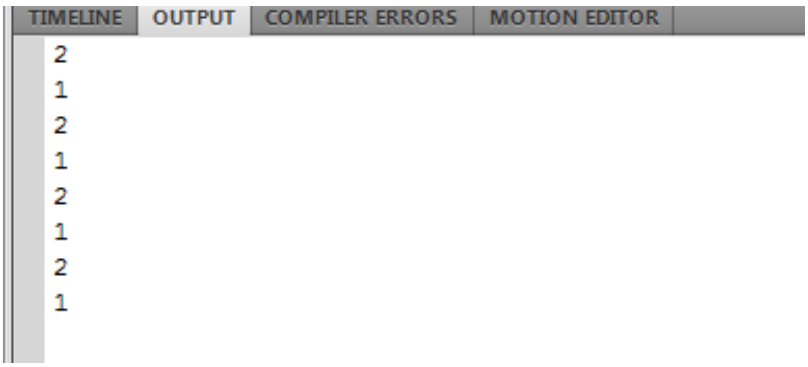
1. En for-loop itereras (upprepas) ett bestämt antal gånger. Se t.ex. följande loop:

```
for (var i:int = 0; i < 5; i++) {
                    trace(i);
}
```

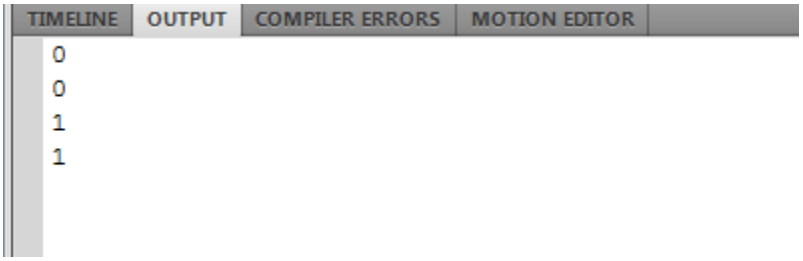
   Här har vi först en iterator (var i:int = 0) som vi initierar till värdet 0. Sedan anger vi villkoret, d.v.s. loopen ska köras så länge **i** är mindre än 5 (i < 5). Slutligen säger vi vad som ska hända (action) för varje loop, d.v.s. att **i** ska öka i värde med 1 (i++).

2. Skapa en ny fla-fil och en ny as-fil och koppla ihop dessa som vanligt. Skapa en init()-metod och kör denna från konstruktorn, precis som vanligt. Lägg in for-loopen från förra steget i init() och testa.
3. Skriv om for-loopen så att den istället för att skriva ut 0-4, skriver ut 4-0 (m.a.o. motsatt resultat från den nuvarande loopen).
4. Skriv om for-loopen så att den skriver ut nummer från -4 till -1.
5. Skriv om for-loopen så att den bara skriver ut jämna nummer mellan 0 och 12.
6. Skriv om for-loopen så den ger följande output (min lösning finns längst ner på nästa sista):

7. Skriv om for-loopen så den ger följande resultat:



8. Skriv om for-loopen så den ger följande resultat (utan att använda två trace()):



9. Gå till fla-filen och skapa en ny symbol (MovieClip) som du döper till Circle. Rita en (ganska liten) cirkel i symbolen. Skriv en for-loop som placerar ut 10 instanser av dessa cirklar (new Circle()) på scenen och slumpar fram deras x- och y-positioner (använd t.ex. Math.random på samma sätt som i övning 3). Min lösning finns på näst sista sidan i detta dokument.

LÖSNING TILL 6:

```
for (var i:int = 0; i <2; i++) {
    for (var j:int = 0; j < 5; j++) {
        trace(j);
    }
}
```

EXTRAÖVNING 2 AV 7 – LADDA IN EXTERNA SWF

För att ladda in externa filer används metoden load() från klassen Loader, vilket i sin tur kräver en URLRequest. Det låter som ganska mycket, men är faktiskt väldigt enkelt.

1. För att ladda in en extern SWF används följande kod:

```
var myLoader:Loader = new Loader();
var url:URLRequest = new URLRequest("MySWF.swf");
myLoader.load(url);
addChild(myLoader);
```

2. Vad är det vi gör?
    a. Vi skapar först en ny Loader (precis som vilken annan ny instans).
    b. Vi skapar en ny URLRequest, där vi anger sökvägen till SWF-filen vi vill ladda in (tänk på att ange exakt sökväg, t.ex. om den ligger i en annan mapp).
    c. Vi använder metoden load() för att ladda vår URLRequest.
    d. Vi lägger till var Loader på scenen – precis som ett vanligt MovieClip eller liknande.

Om du vill vara säker på att filen laddats innan du lägger till den på scenen kan du lägga till en event listener som lyssnar efter när laddningen är klar:

```
myLoader.addEventListener(Event.COMPLETE,  onLoadComplete);
```

Sedan är det bara att lägga till SWF:en på scenen i event handlern istället.

Det finns två andra praktiska event som brukar användas tillsammans med en loader:

- Event.INIT
- ProgressEvent.PROGRESS

Kolla gärna upp hur dessa används.

EXTRAÖVNING 3 AV 7 – ANVÄNDA TIMER

Timers kan vara bra att ha till många olika saker, t.ex. visa hur länge ett spel pågått, sätta en tidsgräns för spelet, eller få någonting att hända med olika intervaller. En timer fungerar ihop med event, och tillsammans kan man använda dessa för en mängd funktionaliteter.

1. Deklarera först ett nytt timer-objekt och importera Timer-klassen (flash.utils.Timer). Timer-konstruktorn kräver två parametrar: vilken fördröjning varje timer-event ska ha, d.v.s. hur ofta den ska avfyras, vilket anges i millisekunder; samt hur många gånger timern ska upprepas (sätts denna parameter till 0 upprepas timern oändligt många gånger).

```
var myTimer:Timer = new Timer(1000, 1); // Fired once, after one second

var myTimer:Timer = new Timer(2000, 0); // Fired forever, each two seconds

var myTimer:Timer = new Timer(1000, 0); // Fired forever, every second

var myTimer:Timer = new Timer(1000, 5); // Fired five times, one second apart
```

2. Lägg sedan till en event listener (på timer-objektet) som lyssnar efter TimerEvent.TIMER (importera flash.events.TimerEvent). Detta event avfyras alltså med det intervall du satt tidigare för timern (i millisekunder).

```
myTimer.addEventListener(TimerEvent.TIMER, onTimerEvent);
```

3. Starta sedan timern:

```
myTimer.start();
```

4. Skriv sedan en event handler som utför det som du vill ska hända vid varje intervall, t.ex. uppdatera tiden i en tidsvisare:

```
myText.text = ”Tid:” + myTimer.currentCount;
```

   Eller flytta ett objekt på scenen:

```
myMovieClip.x += 10;
myMovieClip.y += 10;
```

   Eller kolla om spelet pågått för länge och det är game over:

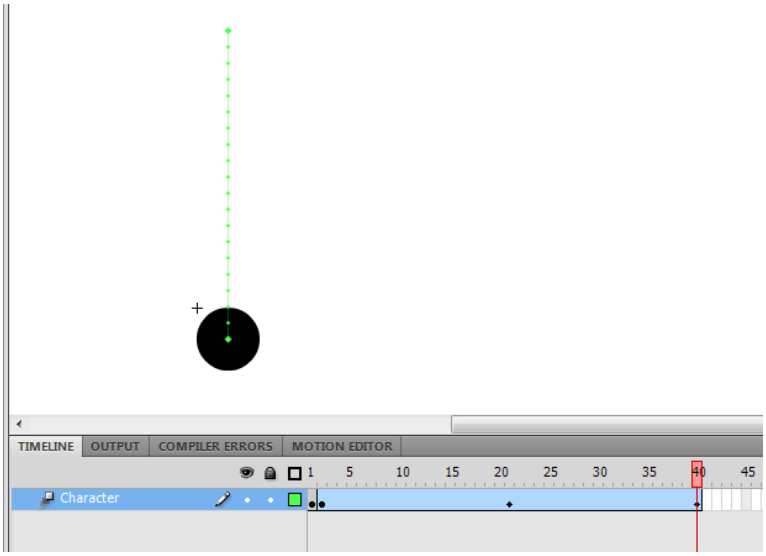
```
var timeLimit:int = 60;
//Recalculates timer to minutes and compares to time limit for game
if(myTimer.currentCount >= timeLimit) {
        gameOver();
}
```

5. Testa att lägga till en tidsvisare som visar hur lång tid som gått på sekundnivå (du behöver lägga till ett textfält på scenen för att visa tiden) med hjälp av en timer. Min lösning finns på sista sidan.

EXTRAÖVNING 4 AV 7 – HOPPA

Ett enkelt sätt att skapa en figur som hoppar vid en knapptryckning är följande:

1. Skapa ett Movie Clip med en figur som är stillastående på frame 1, och från och med frame 2 animeras hoppande (t.ex. med en motion tween). Min väldigt enkla lösning:



2. Använd koden från en tidigare övning (t.ex. del två av övning 4) för att få figuren att röra sig (utan att hoppa, d.v.s. precis som vanligt). Ändra dock koden så att det bara går att röra sig i sidled, inte upp och ner. I init()-metoden kan du lägga till ditt Movie Clip på scenen, samt stoppa det på frame 1.
3. I koden för onKeyDownPress, bygg på den existerande if-satsen för att testa om spelaren tryckt på space, och lägg där till en gotoAndPlay(2) på spelaren.
4. I onEnterFrameEvent, använd en if-sats för att se om animeringen är på sista framen, och lägg där till en gotoAndStop(1) för att få figuren att återvända till ursprungsläget när hoppet är klart.
5. Nu ska figuren hoppa när användaren trycker på space. Testa att göra detta själv. Behöver du hjälp finns min lösning på nästa sida.

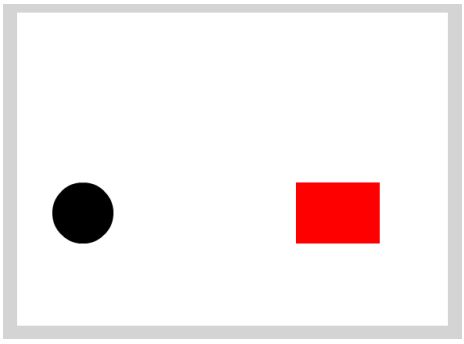
```as3
package  {
    import flash.display.MovieClip;
    import flash.events.KeyboardEvent;
    import flash.ui.Keyboard;
    import flash.events.Event;

    public class Main extends MovieClip {
        //Declare global variables
        var vx:int;
        var vy:int;
        var velocity:uint;
        var player:Player;

        //--------------------
        //Constructor
        public function Main() {
            init();
        }

        //--------------------
        //Inilializes game
        function init():void {
            //Initialize variables
            vx = 0;
            vy = 0;
            velocity = 5;
            player = new Player();
            player.y = stage.stageHeight - player.height;
            player.stop();

            //Adds player to stage
            addChild(player);

            //Add event listeners
            stage.addEventListener(KeyboardEvent.KEY_DOWN, onKeyDownPress);
            stage.addEventListener(KeyboardEvent.KEY_UP, onKeyUpPress);
            stage.addEventListener(Event.ENTER_FRAME, onEnterFrameEvent);
        }

        //--------------------
        //Handles key down press
        function onKeyDownPress(e:KeyboardEvent):void {
            //Checks which button was pressed
            if (e.keyCode == Keyboard.LEFT) {
                vx = -velocity; //Moves player left
            } else if (e.keyCode == Keyboard.RIGHT) {
                vx = velocity; //Moves player right
            } else if (e.keyCode == Keyboard.SPACE) {
                player.gotoAndPlay(2);
            }
        }

        //--------------------
        //Handles key release
        function onKeyUpPress(e:KeyboardEvent):void {
            //Resets variables values depending on key pressed (to stop player)
            if (e.keyCode == Keyboard.LEFT || e.keyCode == Keyboard.RIGHT) {
                vx = 0;
            }
        }

        //--------------------
        //Handles enter frame
        function onEnterFrameEvent(e:Event):void {
            //Move the player (i.e. updates player position)
            player.x += vx;
            player.y += vy;

            //Checks if jump animation is on last frame
            if(player.currentFrame == player.totalFrames) {
                player.gotoAndStop(1); //Goes to 1 and stops
            }

        }

    }

}
```

EXTRAÖVNING 5 AV 7 – SKJUTA

Det finns många olika sätt att göra så att det går att ”skjuta” med AS3.0. Detta är kanske inte det bästa sättet, men det är relativt enkelt. Vi kommer dock behöva använda både en for-loop (du kan träna på att skriva dessa i övning 1 ovan innan du gör denna övning), en typ av event vi aldrig använt, samt en metod vi aldrig använt (getQualifiedClassName), vilka jag kommer att beskriva nedan.

1. Börja med att skapa tre objekt i en fla-fil:
   a. Shooter som är tänkt ska skjuta. Lägg till denna på scenen.
   b. Target som är tänkt att vara måltavlan. Lägg även denna på scenen.
   c. Bullet, d.v.s. en ”kula”. En liten cirkel är nog enklast för denna övning. Denna ska dock inte läggas ut på scenen.

   Min väldigt enkla lösning ser ut så här (svart är shooter, röd är target):



2. I din constructor ska du denna gång inte bara skriva init() som vanligt, utan vi måste försäkra oss om att scenen inte är null (d.v.s. att den hunnit ladda) eftersom vi direkt i init() sedan ska lägga till event listeners på scenen. Detta är mest en försiktighetsåtgärd, och behövs inte alltid, men bör göras. Importera därför flash.events.Event och skriv följande i constructorn:

```
addEventListener(Event.ADDED_TO_STAGE, init);
```

3. Skriv init()-metoden som vanligt, men inom parentesen lägg till e:Event.
4. I init(), lägg till en event listener som lyssnar efter KEY_UP och en som lyssnar efter ENTER_FRAME. Kom ihåg att importera nödvändiga paket. Skriv event handlers för båda event listeners.
5. I event handlern för KEY_UP, använd en if-sats för att se om användaren klickar på space och i så fall lägg till en ny Bullet på scenen, samt ställ in dess x och y –koordinater.
6. I event handlern för ENTER_FRAME skriv en for-loop som går igenom alla scenens barn (använd numChildren för att ta reda på hur många barn scenen har), kolla om barnet är en Bullet (använd getChildAt(i) för att referera till det aktuella barnet), och i så fall förflytta det samt kolla om kulan träffat måltavlan med hitTestObject eller hitTestPoint. Om det är en träff tar du bort kulan från scenen. För att kolla vilket klass en variabel är av (d.v.s. i vårt fall om barnet är en Bullet) kan du använda följande metod:

```
getQualifiedClassName();
```

   Denna metod returnerar en String med namnet på klassen, i vårt fall alltså ”Bullet”. För att använda metoden måste du importera flash.utils.getQualifiedClassName.
7. Nu ska det gå att skjuta, och vid en träff försvinner kulan. Det behövs givetvis mer för att det ska bli ett spel (just nu kan vi ju inte ens röra på vår figur), men det får du bygga själv. Min kod finns på nästa sida.

```as3
package  {
    import flash.display.MovieClip;
    import flash.ui.Keyboard;
    import flash.events.KeyboardEvent;
    import flash.events.Event;
    import flash.utils.getQualifiedClassName;

    public class Main extends MovieClip {

        //--------------------------
        //Constructor
        public function Main() {
            //Listens for when stage is recognised, runs init
            addEventListener(Event.ADDED_TO_STAGE, init);
        }

        //--------------------------
        //Initialises game
        private function init(e:Event):void {
            //Adds event listeners
            stage.addEventListener(KeyboardEvent.KEY_UP, onKeyUpPress);
            stage.addEventListener(Event.ENTER_FRAME, onEnterFrameEvent);
        }

        //--------------------------
        //Handles key down press
        private function onKeyUpPress(e:KeyboardEvent):void {
            //If user has pressed spacebar to shoot
            if(e.keyCode == Keyboard.SPACE) {
                var bullet:Bullet = new Bullet(); //New bullet instance
                bullet.x = shooter.x + shooter.width; //Sets x for new bullet
                bullet.y = shooter.y + shooter.height / 2; //Sets y for new bullet

                stage.addChild(bullet); //Adds bullet to stage
            }
        }

        //--------------------------
        //Handles enter frame event
        private function onEnterFrameEvent(e:Event) {
            //Loops through all the children of the stage
            for(var i:int = 0; i < stage.numChildren; i++) {
                //Checks if the current child is a bullet
                if(getQualifiedClassName(stage.getChildAt(i)) == "Bullet") {
                    stage.getChildAt(i).x += 5; //Moves child

                    //Checks if current child has hit the target
                    if(stage.getChildAt(i).hitTestObject(target)) {
                        trace("Hit!");
                        stage.removeChild(stage.getChildAt(i)); //Removes current child from stage
                    }
                }
            }
        }
    }
}
```

EXTRAÖVNING 6 AV 7 – EGEN MUSPEKARE

Ibland kan det vara önskvärt att använda en egendesignad muspekare istället för den vanliga pilen. Så här går du till väga för att skapa en sådan:

1. Skapa en ny fla-fil och en tillhörande as-fil. Koppla ihop dem som vanligt.
2. I fla-filen, skapa ett nytt movie clip som du döper till MyCursor. Rita där den bild du vill ha som muspekare. Dra inte ut den på scenen.
3. Du får en kodstomme som du kan arbeta utifrån för denna uppgift. Skriv därför in den kod som behövs så att din as-fil ser ut på följande sätt:

```
1  package  {
2      import flash.events.Event;
3      import flash.display.MovieClip;
4
5      public class Main extends MovieClip {
6
7          //--------------------
8          //Constructor
9          public function Main() {
10             addEventListener(Event.ADDED_TO_STAGE, onAddedToStage); //Listens for when stage is loaded
11         }
12
13         //--------------------
14         //Handles when stage has loaded
15         private function onAddedToStage(e:Event):void {
16             init();
17         }
18
19     }
20
21 }
```

4. Det vi gör i koden ovan är att i konstruktorn lägga in en event listener som lyssnar efter när scenen laddats, och den tillhörande event handlern kör i sin tur init(). Detta är för att scenen inte alltid hinner ladda innan du börjar lägga till saker på den, och så kommer du få ett fel.
5. Deklarera en ny global privat variabel (_myCursor) som är av typen MyCursor.
6. Som du ser finns ingen init()-metod än, utan den får du skriva själv. Den bör innehålla följande:
    a. Initiera _myCursor som en ny MyCursor.
    b. Dölj _myCursor (visible = false).

       Att dölja _myCursor är egentligen inte nödvändigt, utan görs bara för att undvika att pekaren hamnar på koordinaterna 0, 0 innan användaren börjar röra på den. Testa gärna att kommentera bort raden och se vad skillnaden blir.

    c. Lägg till koden (Du behöver importera flash.ui.Mouse):

       ```
       Mouse.hide();
       ```

       Denna rad gör så att den vanliga pekaren göms.
    d. Lägg till _myCursor på scenen.
    e. Lägg till en event listener som lyssnar efter MouseEvent.MOUSE_MOVE (importera flash.events.MouseEvents). Detta event avfyras varje gång muspekaren rörs.
7. Skriv event handlern till det event du lade till. I event handlern, lägg till följande:

```
_myCursor.visible = true; //Shows custom cursor
_myCursor.x = e.stageX; //Sets custom cursor x coordinates to regular cursor's
_myCursor.y = e.stageY; //Sets custom cursor y coordinates to regular cursor's
e.updateAfterEvent(); //Makes the cursor move smoother
```

Det vi gör här är att först visa _myCursor, som vi ju dolde i steg 6. De två följande raderna ställer in x- och y-koordinaterna för din egen pekare baserat på var den riktiga muspekaren befinner sig. Vi använder in-parametern till event handlern (**e**) för att ta reda på vilka koordinater muspekaren har (stageX och stageY). Slutligen, på sista raden, använder vi en metod som helt enkelt uppdaterar eventet, och gör att vår egen muspekare rör sig lite smidigare. Om du vill kan du testa att kommentera bort denna rad och se vad skillnaden blir.

8. Slutligen är det två saker till vi bör göra. Lägg till följande rader i init():

```
_myCursor.mouseEnabled = false; //Disables mouse events for custom cursor
_myCursor.mouseChildren = false; //Disables mouse events for custom cursor's children
```

Detta gör att vår nya muspekare fungerar som en vanlig muspekare. Det är inte nödvändigt att ha med dessa om du inte tänker använda funktioner som hover och mouse over och dylikt.

9. Min lösning finns på nästa sida. Försök dock göra övningen utan att kika på den.
10. Testa din applikation. Nu har du din egen muspekare som fungerar!

Notera att för att muspekaren ska fungera måste _myCursor vara det sista barnet som läggs till på scenen. Om du får problem med att använda den egna muspekaren kan det bero på detta.

```as3
package  {
    import flash.ui.Mouse;
    import flash.events.MouseEvent;
    import flash.events.Event;
    import flash.display.MovieClip;

    public class Main extends MovieClip {

        //Global variables
        private var _myCursor:MyCursor;

        //--------------------
        //Constructor
        public function Main() {
            addEventListener(Event.ADDED_TO_STAGE, onAddedToStage); //Listens for when stage is loaded
        }

        //--------------------
        //Initialises app
        private function init():void {
            _myCursor = new MyCursor(); //Creates new custom cursor
            _myCursor.visible = false; //Hides custom cursor
            _myCursor.mouseEnabled = false; //Disables mouse events for custom cursor
            _myCursor.mouseChildren = false; //Disables mouse events for custom cursor's children

            Mouse.hide(); //Hides existing cursor

            stage.addChild(_myCursor); //Must be added LAST so it's on top of the display list!
            /*
            Notera att _myCursor måste vara den senaste addChild som läggs till eftersom den måste ligga "övers"
            i display list. Stöter du på problem med att pekaren slutar röra på sig så kan det bero på att du gjort
            en addChild() efter att du lade till pekaren.
            */

            stage.addEventListener(MouseEvent.MOUSE_MOVE, onMouseMoveEvent); //Listens for mouse movement
        }

        //--------------------
        //Handles when stage has loaded
        private function onAddedToStage(e:Event):void {
            init();
        }

        //--------------------
        //Handles mouse move event
        private function onMouseMoveEvent(e:MouseEvent):void {
            _myCursor.visible = true; //Shows custom cursor
            _myCursor.x = e.stageX; //Sets custom cursor x coordinates to regular cursor's
            _myCursor.y = e.stageY; //Sets custom cursor y coordinates to regular cursor's
            e.updateAfterEvent(); //Makes the cursor move smoother
        }

    }

}
```

EXTRAÖVNING 7 AV 7 – PRELOADER

En preloader kan användas för att visa hur mycket av en fil (din swf, i det här fallet) som har laddats, istället för att användaren ska behöva vänta vid en blank skärm. Gör följande:

1. Börja med att skapa en ny fla-fil och tillhörande as-fil. Koppla ihop dem som vanligt.
2. Skapa ett nytt movie clip som du döper till Preloader. Här ska du rita en rektangel med en 3 px ram, precis som du gjorde hälsomätaren i övning 4 (se även här till att Object drawing mode inte är på – undrar du över detta kan du kika på sista delen av övning 4).
3. Markera bara fyllnaden (så att den blir prickig) och gör om denna till ett Movie Clip. Döp den till Loadbar och ge den instansnamnet loadbar.
4. Dra ut Preloader på scenen i frame 1 och ge den instansnamnet preloader. Se till att det du vill visa efter preloadern inte ligger på frame 1 utan börjar först på frame 2.
5. Gå till as-filen och skriv metoden init() som du kör från konstruktorn.
6. I init, lägg till en event listener som lyssnar efter Event.ENTER_FRAME (glöm inte att importera flash.events.Event).
7. Skriv event handlern för ENTER_FRAME eventet. I denna, lägg till följande:

```
var total:Number = this.stage.loaderInfo.bytesTotal;
var loaded:Number = this.stage.loaderInfo.bytesLoaded;

if(loaded < total) {
	preloader.loadbar.scaleX = loaded/total;
} else {
	gotoAndStop(2);
}
```

   Det vi gör är följande:
   a. Deklarerar två nya variabler (total och loaded) som innehåller info om hur många bytes scenen har, samt hur många av dessa bytes har laddat.
   b. Vi använder sedan en if-sats för att så länge som antalet laddade bytes är mindre än den totala storleken uppdatera vår preloader.
   c. När alla bytes laddats går vi till frame 2 (i else-satsen). Här kanske du vill byta ut gotoAndStop mot gotoAndPlay – eller något annat som passar din applikation bättre.
8. Om du skulle testa preloadern med Ctrl + enter nu skulle den bara blinka förbi, innan Flash går vidare till frame 2. Det är för att när vi testar lokalt finns ju redan alla filer laddade på datorn. Det går dock att köra en ”Simulate download” som kan vara bra i det här fallet. När du klickat på Ctrl + enter och din applikation visas, klicka då på Ctrl + enter igen. Nu simulerar vi en nedladdning, och preloadern har chans att visas.

   Notera dock att om du har en väldigt liten applikation så kommer preloadern ändå bara visas väldigt snabbt, eftersom det laddas färdigt väldigt snabbt. För att testa, lägg t.ex. in en stor ljudfil på frame 2, så kan du se att preloadern ökar långsamt när du simulerar nedladdningen.
9. Testa att göra övningen själv. Min färdiga kod finns på nästa sida om du behöver hjälp.

```
package  {
    import flash.display.MovieClip;
    import flash.events.Event;

    public class Main extends MovieClip {

        //---------------------
        //Constructor
        public function Main() {
            init();
        }

        //---------------------
        //Initialises
        private function init():void {
            //Listens for enter frame
            stage.addEventListener(Event.ENTER_FRAME, onEnterFrameEvent);
        }

        //---------------------
        //Handles enter fram event
        private function onEnterFrameEvent(e:Event):void {
            var total:Number = this.stage.loaderInfo.bytesTotal; //Total no. of bytes in app
            var loaded:Number = this.stage.loaderInfo.bytesLoaded; //Amount of loaded bytes

            //If not loaded, increases preloader bar
            if(loaded < total) {
                preloader.loadbar.scaleX = loaded/total; //Sets preloader bar size
            } else {
                gotoAndStop(2); //Continues app
            }

        }

    }

}
```

LÄNKTIPS

Följande länkar har bra information om ActionScript 3.0, och kan vara till hjälp när du lär dig programmera. Kolla främst på Adobes egna länkar som är mycket bra. Lycka till!

En bra e-bok om AS3.0 (på engelska):
http://help.adobe.com/en_US/ActionScript/3.0_ProgrammingAS3/flash_as3_programming.pdf

Samma bok på svenska:
http://help.adobe.com/sv_SE/ActionScript/3.0_ProgrammingAS3/flash_as3_programming.pdf

Instruktioner om att deklarera variabler:
http://help.adobe.com/en_US/ActionScript/3.0_ProgrammingAS3/WS5b3ccc516d4fbf351e63e3d118a9b90204-7f9d.html

Bra information om grunderna i AS3.0:
http://help.adobe.com/en_US/ActionScript/3.0_ProgrammingAS3/WS5b3ccc516d4fbf351e63e3d118a9b90204-7ec7.html

Om att skriva egna funktioner:
http://help.adobe.com/en_US/ActionScript/3.0_ProgrammingAS3/WS5b3ccc516d4fbf351e63e3d118a9b90204-7f57.html

Inte från Adobe, men bra ändå:

Videotutorials (se till att filtrera på 3.0):
http://www.lynda.com/software/148-actionscript

LÖSNING FOR-LOOP

Lösning på 9:

```
//Places 10 circles on stage with random coordinates
for (var i:int = 0; i < 10; i++) {
    var circle:Circle = new Circle();
    circle.x = Math.ceil(Math.random() * stage.stageWidth); //Random number max as stage width
    circle.y = Math.ceil(Math.random() * stage.stageHeight); //Random number max as stage height
    addChild(circle);
}
```

LÖSNING TIMER

Jag har gjort en lite mer avancerad lösning än vad som egentligen behövs, eftersom jag ville att min tid skulle visas i ett specifikt format med minuter och sekunder (00:00). Det räcker om du gjort init()-metoden samt event handlern, och där lagt till en kod liknande den jag kommenterat bort på rad 29. Du får gärna återanvända min kod för att formatera tiden, om du vill. Det finns dock en mängd olika sätt att lösa detta på som kanske passar din kod bättre.

```
1  package  {
2      import flash.utils.Timer;
3      import flash.events.TimerEvent;
4      import flash.display.MovieClip;
5
6      public class Main extends MovieClip {
7
8          //Global variables
9          private var _timer:Timer;
10
11         //---------------------------------
12         //Constructor
13         public function Main() {
14             init();
15         }
16
17         //---------------------------------
18         //Initialises timer
19         public function init():void {
20             _timer = new Timer(1000, 0); //Fires every second, forever
21             _timer.addEventListener(TimerEvent.TIMER, onTimerEvent);
22             _timer.start(); //Starts timer
23         }
24
25         //---------------------------------
26         //Handles timer event
27         public function onTimerEvent(e:TimerEvent):void {
28             //Changes text to display current time
29             //timeDisplay.text = String(timer.currentCount); //Basic display of number of seconds
30             timeDisplay.text = toFormattedTime(_timer.currentCount); //Advanced formatted display of time
31         }
32
33         //---------------------------------
34         //Formats seconds to 00:00 string
35         public function toFormattedTime(time:Number):String {
36             var minutes:int;
37             var seconds:int;
38
39             minutes = Math.floor(time / 60); //Calculates minutes passed
40             seconds = Math.floor(time % 60); //Calculates remaining seconds (% is the remainder operator)
41
42             //Returns formatted string
43             return toDoubleDigits(minutes) + ":" + toDoubleDigits(seconds);
44         }
45
46         //---------------------------------
47         //Formats number to two digit string
48         public function toDoubleDigits(num:int):String {
49             if (num < 10) {
50                 return "0" + num;
51             }
52             return String(num);
53         }
54     }
55 }
56
```