ÖVNING 5

ÖVNING 1 AV 7 – KROCKA

Den här övningen kommer även den att beröra kontroll av figurer med piltangenter, men denna gång ska vi träna på att plocka upp och släppa objekt, samt hantera kollisioner.

1. Öppna filerna du förberedde i förra övningen: interactivePlayground.fla och Main_Playground.as.
2. I Main_Playground.as, klistra in följande (det är samma kod som vi använde i en av de tidigare övningarna):

```
package
{
	import flash.display.MovieClip;
	import flash.events.KeyboardEvent;
	import flash.ui.Keyboard;
	import flash.events.Event;
	import flash.text.TextField; //obs importera denna classen annars fungerar inte din dynamiska text

	public class Main_Playground extends MovieClip
	{
		var vx:int;
		var vy:int;

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

		//--------------------
		//Initializes game
		private function init():void {
			//Stops animation
			enemy.stop();
			player.stop();

			//initialize variables
			vx = 0;
			vy = 0;

			//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
		private function onKeyDownPress(e:KeyboardEvent):void {
			if (e.keyCode == Keyboard.LEFT) {
				vx = -5;
			} else if (e.keyCode == Keyboard.RIGHT) {
				vx = 5;
			} else if (e.keyCode == Keyboard.UP) {
				vy = -5;
			} else if (e.keyCode == Keyboard.DOWN) {
				vy = 5;
			}
		}

		//--------------------
		//Handles key release
		private function onKeyUpPress(e:KeyboardEvent):void {
			if (e.keyCode == Keyboard.LEFT || e.keyCode == Keyboard.RIGHT){
				vx = 0;
			} else if (e.keyCode == Keyboard.DOWN || e.keyCode == Keyboard.UP) {
				vy = 0;
			}
		}

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

3. Som du ser använder vi ingen blockering för scenen, utan vi håller koden så enkelt som möjligt i det här stadiet. Spara och testa applikationen. Koll igenom koden och se till att du förstår allt. Om du undrar över något kan du återgå till förra övningen och repetera det.
4. Nu ska vi göra så att någonting händer när din figur krockar med fienden, s.k. collision detection. I AS finns en väldigt behändig metod för detta: hitTestObject(). Metoden kollar om

två objekt överlappar varandra. Metoden returnerar en boolean, vilket gör den lämplig att använda i en if-sats. Lägg därför till följande kod i din onEnterFrameEvent():

```
//Collision detection
if (player.hitTestObject(enemy)) {
	messageDisplay.text = "Aj!!";
} else {
	messageDisplay.text = "Ingen krock...";
}
```

5. Spara och testa koden. Som du ser ändras texten när du krockar med fienden.
6. Gör så att både din figur och fienden animeras (frame 2 visas) när en krock sker. Behöver du hjälp på traven finns min lösning på nästa sida.

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

    //Collision detection
    if (player.hitTestObject(enemy)) {
        messageDisplay.text = "Aj!";

        //Animates characters
        player.gotoAndStop(2);
        enemy.gotoAndStop(2);
    } else {
        messageDisplay.text = "Ingen krock...";

        //Animates characters
        player.gotoAndStop(1);
        enemy.gotoAndStop(1);
    }
}
```

ÖVNING 2 AV 7 – HÄLSOMÄTARE

Vi ska nu få hälsomätaren att fungera. Tanken är att för varje gång figuren krockar med fienden så ska hälsan minska.

1. För att få tillgång till mätaren (som du gav instansnamnet *meter*, vilken ligger i symbolen Health med instansnamnet *health* – kommer du ihåg?) kan du använda dot-notation. Vi vill ändra bredden på mätaren, d.v.s. korta av den.

```
health.meter.width--;
```

2. Lägg in koden på lämpligt ställe och testa din applikation. Vill du att mätaren ska krympa snabbare kan du minska med mer än bara 1 varje gång.
3. Det finns dock även ett annat sätt att göra detta på, vilket kräver mindre datorkraft och därför är den föredragna metoden. Ersätt därför koden från 1 med:

```
if (health.meter.scaleX > 0) {
        health.meter.scaleX -= 0.02;
}
```

4. Här använder vi scaleX istället för width, vilket innebär att vi sätter den horisontella skalan för mätaren. Detta sker med procent, d.v.s. 100 % är 1, vilket är anledningen till att vi minskar med -0.02. If-satsen behövs eftersom om vi inte stoppar minskningen kommer mätaren att börja växa åt vänster istället för att stanna när den nått botten. Du kan testa att ta bort if-satsen och se vad skillnaden blir, om du vill.

5. Än så länge händer ingenting om mätaren når botten. Lägg därför även till följande kod på lämpligt ställe:

```
//Check for end of game
if (health.meter.width < 1) {
	messageDisplay.text = "Game Over!";
}
```

6. Nu skrivs ett meddelande ut när spelet är slut. Det går dock att fortfarande röra på figuren. Detta kommer vi att återkomma till om en liten stund.

ÖVNING 3 AV 7 – POÄNG

1. För att kunna räkna poäng behöver du deklarera en ny variabel: score:uint. Initiera den till värdet 0.
2. Lägg till följande i if-satsen som hanterar krock med fienden (d.v.s. poängen ska öka när figuren krockar med fienden):

```
score++;
scoreDisplay.text = String(score);
```

3. Spara och testa. Poängen uppdateras, men alldeles för ofta. Eftersom vi lagt det i onEnterFrameEvent() så kommer den att uppdateras för varje ny frame, kommer du ihåg? Därför måste vi lägga in en begränsning. Börja med att deklarera en boolean (döp den till collision) och initiera den som false. Ersätt koden från steg 2 med följande:

```
//Increases score
if (!collision) {
	score++;
	scoreDisplay.text = String(score);
	collision = true;
}
```

4. I den else-sats som hanterar om ingen krock sker, lägg till:

```
collision = false;
```

5. Spara och testa. Nu uppdateras bara poängen en gång för varje krock. Vad vi gjorde var följande:
    a. Vi använder en boolean för att kolla om en krock är pågående. Om vi krockar och en krock inte är pågående (collision är false) ökar vi poängen och skriver ut den. Vi sätter dessutom collision till true eftersom nu är ju en krock pågående.
    b. När vi sedan slutar krocka med fienden måste vi återställa collision till false, och då gör vi det även möjligt att öka poängen igen.
6. Se till att du förstått logiken i det vi gjort innan du fortsätter. Om du inte är med på hur det fungerar, testa att ändra de olika variablerna, eller att lägga in en trace() om det är någon variabel du är osäker på.
7. Vi vill att när en spelare nått 5 poäng ska denne ha vunnit spelet. Lägg in kod som skriver ut att spelaren vunnit när denne fått 5 poäng. Om du är osäker på hur och var du bör göra detta kan du se min färdiga kod på nästa sida.

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

    public class Main_Playground extends MovieClip
    {
        var vx:int;
        var vy:int;
        var score:uint;
        var collision:Boolean;

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

        //--------------------
        //Initializes game
        private function init():void {
            //Stops animation
            enemy.stop();
            player.stop();

            //Initialize variables
            vx = 0;
            vy = 0;
            score = 0;
            collision = false;

            //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
        private function onKeyDownPress(e:KeyboardEvent):void {
            if (e.keyCode == Keyboard.LEFT) {
                vx = -5;
            } else if (e.keyCode == Keyboard.RIGHT) {
                vx = 5;
            } else if (e.keyCode == Keyboard.UP) {
                vy = -5;
            } else if (e.keyCode == Keyboard.DOWN) {
                vy = 5;
            }
        }

        //---------------------
        //Handles key release
        private function onKeyUpPress(e:KeyboardEvent):void {
            if (e.keyCode == Keyboard.LEFT || e.keyCode == Keyboard.RIGHT){
                vx = 0;
            } else if (e.keyCode == Keyboard.DOWN || e.keyCode == Keyboard.UP) {
                vy = 0;
            }
        }

```

```
83          //----------------------
84          //Handles enter frame
85          private function onEnterFrame(e:Event):void {
86              //Move the player
87              player.x += vx;
88              player.y += vy;
89
90              //Collision detection
91              if (player.hitTestObject(enemy)) {
92                  messageDisplay.text = "Aj!";
93
94                  //Reduces health
95                  if (health.meter.scaleX > 0) {
96                      health.meter.scaleX -= 0.02;
97                  }
98
99                  //Increases score
100                 if (!collision) {
101                     score++;
102                     scoreDisplay.text = String(score);
103                     collision = true;
104                 }
105
106                 //Animates characters
107                 player.gotoAndStop(2);
108                 enemy.gotoAndStop(2);
109             } else {
110                 messageDisplay.text = "Ingen krock...";
111
112                 //Animates characters
113                 player.gotoAndStop(1);
114                 enemy.gotoAndStop(1);
115
116                 collision = false;
117             }
118
119             //Check for end of game
120             if (health.meter.width < 1) {
121                 messageDisplay.text = "Game Over!";
122             }
123
124             if (score >= 5) {
125                 messageDisplay.text = "Du vann!";
126             }
127         }
128     }
129 }
130
```

ÖVNING 4 AV 7 – PLOCKA UPP OBJEKT

1. Du ska nu göra så att det går att plocka upp äpplet. Först måste du deklarera en boolean (hasApple), vilken du ger värdet false.
2. Tanken är att du ska plocka upp och släppa äpplet med mellanslagstangenten. Koden för att göra detta ska vi lägga i onKeyDownPress(). Koden ser ut som följer:

```
if (e.keyCode == Keyboard.SPACE && player.hitTestObject(apple)) {
        if (!hasApple) {
                        player.addChild(apple);
                        apple.x = 0;
                        apple.y = 0;
                        hasApple = true;
        } else {
                        stage.addChild(apple);
                        apple.x = player.x;
                        apple.y = player.y;
                        hasApple = false;
        }
}
```

   Lägg till koden på lämpligt ställe i din onKeyDownPress().

3. Spara och testa. Om du nu ställer dig vid äpplet och trycker på space så läggs äpplet på din figur, och du kan flytta det med dig. Tryck på mellanslag och du släpper äpplet igen. Vad vi gjort är följande:
   a. Vi lyssnar efter om användaren trycker på space samtidigt som figuren ”krockar” med äpplet.
   b. Om vi inte har äpplet (d.v.s. hasApple är false) så plockar vi upp det genom att lägga till det som ett ”barn” till figuren (kommer du ihåg hur display list fungerade? Här ser du hur man kan utnyttja det). Vi sätter äpplets koordinater till 0, vilket anpassar äpplet till dess förälders koordinater – i det här fallet alltså figuren.
   c. Vi sätter hasApple till true så att vi vet att figuren just nu bär på äpplet.
   d. Om vi däremot redan har äpplet lägger vi till det på scenen igen (istället för att ha den som ”barn” till figuren). Äpplet får samma koordinater som figuren på scenen, och vi sätter hasApple till false så att vi vet att figuren inte längre har äpplet.

ÖVNING 5 AV 7 – GAME OVER

Nu ska vi (äntligen) göra så att spelet verkligen tar slut när hälsomätaren når botten. Kommer du ihåg knappen ReplayButton som du skapade i förra övningen? Den ska vi använda för att låta användaren starta om spelet när det tagit slut. Dock har vi ju inte lagt till det på scenen, så vi måste gå till väga lite annorlunda än vanligt.

1. Deklarera en variabel (replayButton) av typen ReplayButton. Initiera den till en ny ReplayButton() (om du kommer ihåg så gjorde vi något liknande med våra Movie Clip i interactivePlayground. Hint: använd nyckelordet new).
2. Vi vill att spelet ska sluta fungera både när spelet vunnits och förlorats, vilket innebär att en enkel lösning är att skriva en ny metod (som vi kan kalla endGame()) som hanterar det vi vill göra i båda situationerna (då kan vi återanvända metoden). Fundera över vad som kan vara bra att ha med i en sådan metod. Det vi behöver göra är:
    a. Importera MouseEvent.
    b. Lägga till replay-knappen på scenen.
    c. Lägga till en event listener som lyssnar efter om användaren trycker på knappen för att starta om.
    d. Stoppa spelet genom att ta bort de event listeners som hanterar spelet.
    e. Återställ hälsomätaren (på lämpligt ställe).
    f. Om en krock är pågående är det nog bäst att flytta figuren en bit bort från fienden så att spelet startar ”nollställt”.
    g. Skriva en event handler som startar om spelet när spelaren klickar på replay-knappen.
3. Testa att göra dessa ändringar själv. Fastnar du, eller behöver du hjälp på traven, så finns min lösning på nästa sida.

```actionscript
package
{
    import flash.display.MovieClip;
    import flash.events.KeyboardEvent;
    import flash.ui.Keyboard;
    import flash.events.Event;
    import flash.text.TextField;
    import flash.events.MouseEvent;

    public class Main_Playground extends MovieClip
    {
        var vx:int;
        var vy:int;
        var score:uint;
        var collision:Boolean;
        var hasApple:Boolean;
        var replayButton:ReplayButton;

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

        //--------------------
        //Initializes game
        private function init():void {
            //Stops animation
            enemy.stop();
            player.stop();

            //Initialize variables
            vx = 0;
            vy = 0;
            score = 0;
            collision = false;
            hasApple = false;
            replayButton = new ReplayButton();

            //Sets score message
            scoreDisplay.text = String(score);

            //Sets replay button coordinates
            replayButton.x = 300;
            replayButton.y = 200;

            //Resets health meter
            health.meter.scaleX = 1;

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

            //Sets focus to stage (needed after replay)
            stage.focus = stage;
        }
```

```
59          //----------------------
60          //Handles end of game
61          private function endGame():void {
62              //Adds replay button to stage
63              stage.addChild(replayButton);
64
65              //Adds listener to replay button
66              replayButton.addEventListener(MouseEvent.CLICK, onReplayButtonClick);
67
68              //Removes event listeners for game play
69              stage.removeEventListener(KeyboardEvent.KEY_DOWN, onKeyDownPress);
70              stage.removeEventListener(KeyboardEvent.KEY_UP, onKeyUpPress);
71              stage.removeEventListener(Event.ENTER_FRAME, onEnterFrameEvent);
72          }
73
74          //--------------------
75          //Handles replay button click
76          private function onReplayButtonClick(e:MouseEvent) {
77              //Removes replay listener
78              replayButton.removeEventListener(MouseEvent.CLICK, onReplayButtonClick);
79
80              //Removes replay button
81              stage.removeChild(replayButton);
82
83              //Moves player in case of ongoing collision
84              if (player.hitTestObject(enemy)) {
85                  player.x -= 50;
86              }
87
88              //Restarts game
89              init();
90          }
92          //--------------------
93          //Handles key down press
94          private function onKeyDownPress(e:KeyboardEvent):void {
95
96              if (e.keyCode == Keyboard.LEFT) {
97                  vx = -5;
98              } else if (e.keyCode == Keyboard.RIGHT) {
99                  vx = 5;
100             } else if (e.keyCode == Keyboard.UP) {
101                 vy = -5;
102             } else if (e.keyCode == Keyboard.DOWN) {
103                 vy = 5;
104             } else if (e.keyCode == Keyboard.SPACE && player.hitTestObject(apple)) {
105                 if (!hasApple) {
106                     player.addChild(apple);
107                     apple.x = 0;
108                     apple.y = 0;
109                     hasApple = true;
110                 } else {
111                     stage.addChild(apple);
112                     apple.x = player.x;
113                     apple.y = player.y;
114                     hasApple = false;
115                 }
116             }
117         }
118
119         //----------------------
120         //Handles key release
121         private function onKeyUpPress(e:KeyboardEvent):void {
122             if (e.keyCode == Keyboard.LEFT || e.keyCode == Keyboard.RIGHT){
123                 vx = 0;
124             } else if (e.keyCode == Keyboard.DOWN || e.keyCode == Keyboard.UP) {
125                 vy = 0;
126             }
127         }
```

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

            //Collision detection
            if (player.hitTestObject(enemy)) {
                messageDisplay.text = "Aj!";

                //Reduces health
                if (health.meter.scaleX > 0) {
                    health.meter.scaleX -= 0.02;
                }

                //Increases score
                if (!collision) {
                    score++;
                    scoreDisplay.text = String(score);
                    collision = true;
                }

                //Animates characters
                player.gotoAndStop(2);
                enemy.gotoAndStop(2);
            } else {
                messageDisplay.text = "Ingen krock...";

                //Animates characters
                player.gotoAndStop(1);
                enemy.gotoAndStop(1);

                collision = false;
            }

            //Check for end of game
            if (health.meter.width < 1) {
                messageDisplay.text = "Game Over!";
                endGame();
            }

            if (score >= 5) {
                messageDisplay.text = "Du vann!";
                endGame();
            }
        }
    }
}
```

ÖVNING 6 AV 7 – BEGRÄNSA OMGIVNINGEN

Vi ska nu använda oss av hitTestPoint()för att förhindra att din figur lämnar marken, d.v.s. hålla den inom det gröna området.

1. Lägg till följande inom din onEnterFrameEvent():

```
//hitTestPoint example
if (hill.hitTestPoint(player.x, player.y, true)) {
        trace("On hill");
} else {
        trace("Not on hill.");
}
```

2. Spara och testa. Eftersom vi lagt det i onEnterFrameEvent() kommer våra trace() att skriva ut texten väldigt många gånger, men det går att se om det fungerar eller inte. När du nu rör figuren utanför kullen så kommer ”Not on hill” att skrivas ut.
3. Det enda som är nytt här är raden:

```
hill.hitTestPoint(player.x, player.y, true)
```

4. Tidigare har vi använt hitTestObject(), och hitTestPoint() liknar den. Dock tar hitTestPoint() tre parametrar istället för en:
    a. X-koordinaten av punkten
    b. Y-koordinaten av punkten
    c. Boolean som säger om du vill använda själva formen av objektet eller den omgivande ramen (t.ex. om du har en rund figur vill du bara räkna det som syns).
5. Skillnaden mellan hitTestObject() och hitTestPoint() är att den sistnämnda bara jämför med en specifik punkt, medan den första jämför med hela objektet. Du kan testa att skriva en hitTestObject() istället och se vad skillnaden blir.
6. Vi vill som sagt förhindra att figuren lämnar gräset, så du kan nu ersätta koden från 1 med följande:

```
//Stops player from leaving grass
if(!hill.hitTestPoint(player.x, player.y, true)) {
        player.x -= vx;
        player.y -= vy;
}
```

När du använder hitTestPoint() samtidigt som du har en figur där du ställt x och y till t.ex. -25, vilket vi gjorde med förra övningens figur, kan du behöva räkna om x och y för den punkt du jämför med. Med andra ord, istället för att skriva bara player.x kan du behöva skriva (player.x + (player.width / 2)), och motsvarande för y. Testa dig fram.

7. Nu stannar figuren innan den helt lämnar gräset. Som du ser är det inget nytt i koden, men känner du dig osäker på hur det fungerar kan du testa att ändra de olika variablerna för att se vad som händer.

ÖVNING 7 AV 7 – HINDER

1. Slutligen ska vi göra så att ett objekt hindrar figuren att röra sig vidare. Börja med att i interactivePlayground.fla dra ut det Movie Clip du skapade i förra övningen och döpte till Wall. Ge det instansnamnet wall.
2. Skriv en kod som, med hjälp av hitTestObject(), hindrar figuren från att passera ”väggen”, precis som vi gjorde i föregående övning med gräset. Min lösning finns på nästa sida.

```
180
181             //Stops player from passing wall
182             if (player.hitTestObject(wall)) {
183                 player.x-= vx;
184                 player.y-= vy;
185             }
```