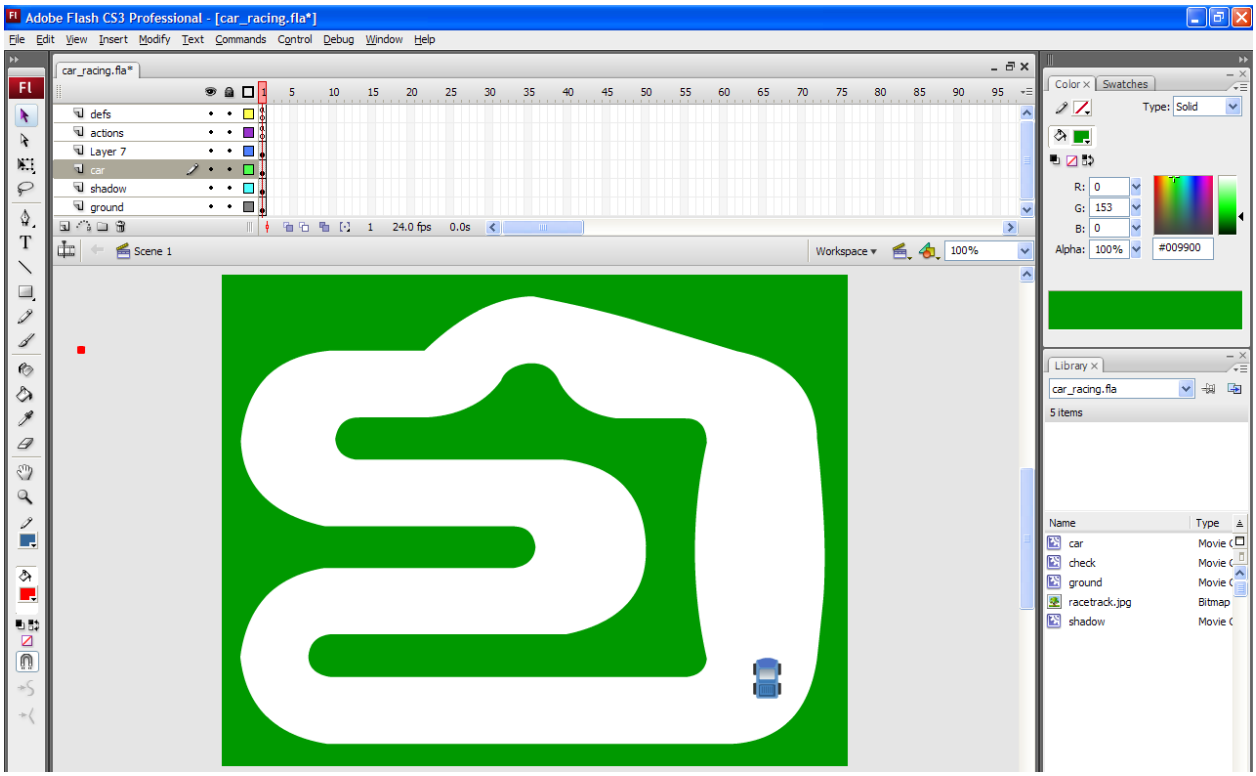
Interaktivna multimedija, drugi sklop vaj

Naloga 1 – sprememba proge

Na internetu smo našli enostavno dirkaško igrico, ki bi jo radi nadgradili. Na dirkaško progo je postavljen avto, ki ga upravljamo s smernimi tipkami.

Za začetek bomo spremenili progo po kateri vozimo avto.

V Flashu odpremo datoteko car_racing.fla, ki se nahaja v direktoriju /vaja2-1. Pri tem lahko opazimo, da za razliko od prve vaje, tokrat v direktoriju nimamo nobene datoteke s končnico **.as**.

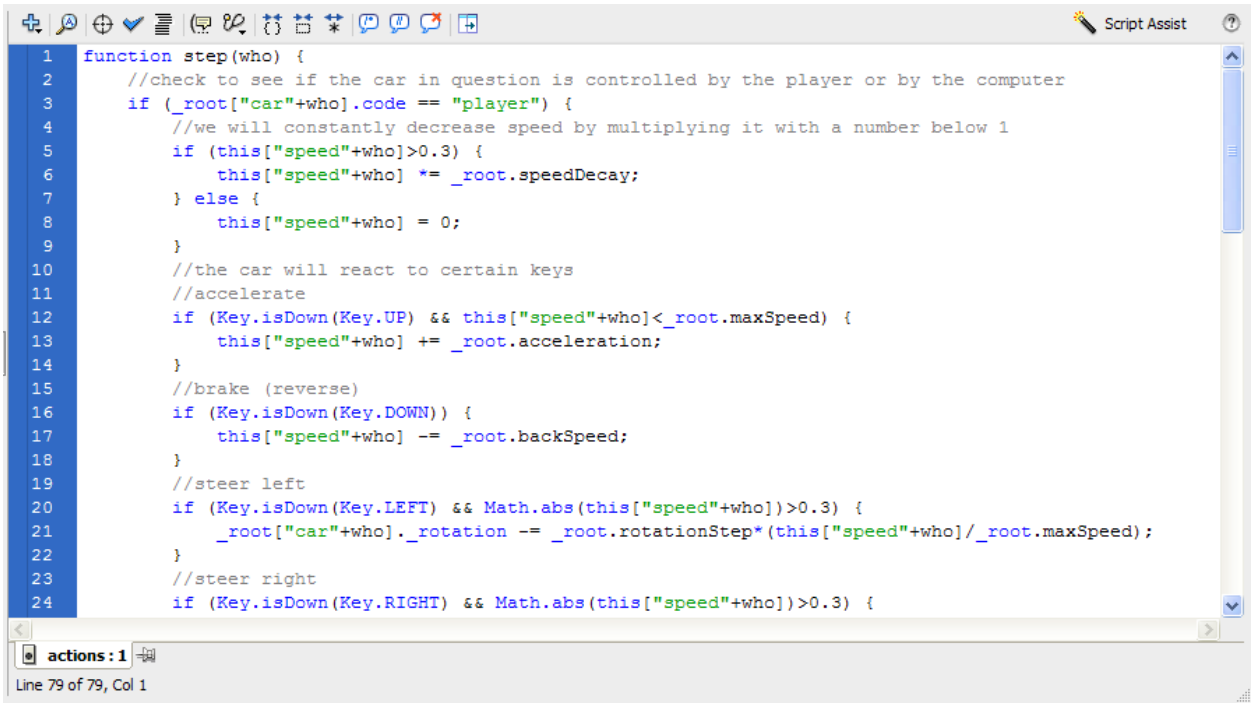


Odpre se nam podobno okno kot zgoraj. V Timeline-u vidimo 6 Layer-jev, v knjižnici (Library) so uporabljeni objekti in grafični elementi, na Stage-u pa je naš avto postavljen na progo. **Zaženemo igro s klikom na CTRL+ENTER in upravljamo avto s pomočjo smernih tipk na tipkovnici**.

Vidimo, da je igra še v začetni stopnji izdelave, saj ni označena štartna črta, ne šteje se čas ali krogi in na progi smo sami. Edino, kar deluje je upravljanje avta in zaznavanje kontakta med avtom in travo izven proge.

Takoj na začetku smo videli, da delovni direktorij ne vsebuje .as datotek, kar je nenavadno glede na to, da smo v prvi vaji vso kodo razredov imeli prav v teh datotekah. V tej igri pa .as datotek ni, ker je vsa koda zapisana v Keyframe-ih na Timeline-u, v obliki takoimenovanih akcij (Actions). Namreč, Timeline v Flashu se v osnovi obnaša kot časovna os v filmu - skozi čas se premika po sličicah (Framih), v sličicah so posamezni koraki animacij in njihovo hitro spreminjanje ustvari občutek zveznega premikanja likov na sličicah. Če pa določene sličice označimo za ključne (Keyframes), potem lahko v njih vpisujemo kodo, ki lahko prekine linearen časovni tok na časovni osi in na ta način naredi našo Flash animacijo interaktivno. Osnovni ukazi v akcijah so stop, play, gotoAndPlay itd. Akcije (koda) bodo delovale ne glede na to v kateri Layer jih postavimo, vendar je zaradi preglednosti najbolje, da naredimo kar Layer Actions, ki vsebuje samo kodo, brez grafičnih elementov.

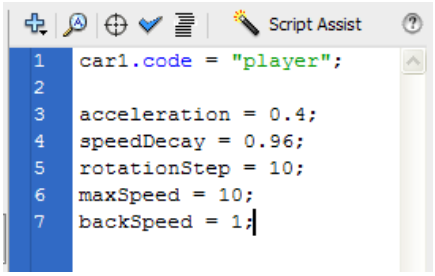
Tudi naš Timeline vsebuje Layer Actions. Kliknemo na edini Frame na Layerju Actions in stisnemo tipko F9. Odpre se nam novo podokno, ki vsebuje kodo za ta Frame (ki je trenutno tudi edini). Če nato klikamo na ostale Layerje v Timeline-u, bomo videli da ne vsebujejo kode, razen Layerja **Defs**, ki pa vsebuje začetne vrednosti spremenljivk v igri, kot npr. hitrost in podobno.



Koda se lahko nahaja tudi kar v objektih iz razreda MovieClip, v našem primeru v avtu. **Kliknemo na avto, da se prikaže moder rob in nato stisnemo tipko F9**. Odpre se nam novo podokno z akcijami MovieClipa.

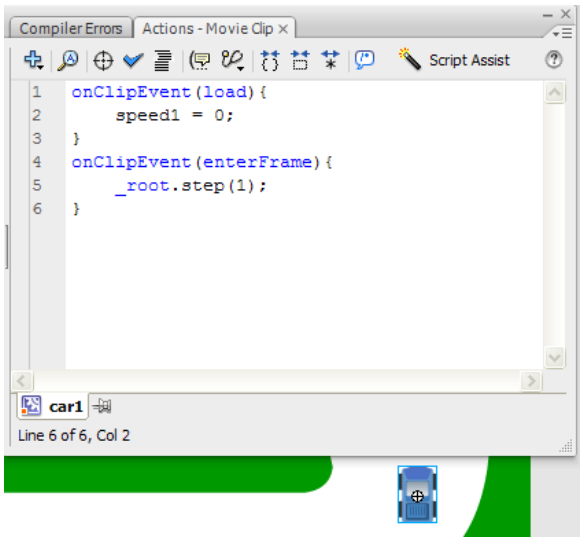
Analiza kode v igrici

Preden zamenjamo ozadje, moramo vsaj približno razumeti kako igrica deluje.



Začetne spremenljivke se nahajajo v Layerju **Defs,** podobno kot v prvi vaji v funkciji OnLoad(). Določene so osnovne spremenljivke, kot koda avta, pospešek in pojemek, hitrost obračanja, maksimalna hitrost in vzvratna hitrost. Koda avta služi zato, da lahko kasneje dodamo več avtov.

Naslednji kos kode se nahaja v MovieClipu **car1**. Znotraj sta samo dva Event listenerja, ki sta pravzaprav ekvivalentna funkcijam OnLoad() in OnEnterFrame() iz prve vaje. V Eventu **load** (ko se avto pojavi na sceni) se postavi začetna hitrost avta na 0, v eventu **enterFrame** (torej v vsakem Frame-u) pa se izvaja funkcija **step()**, ki se nahaja na časovni osi (Timeline) _root, torej na glavni časovni osi.



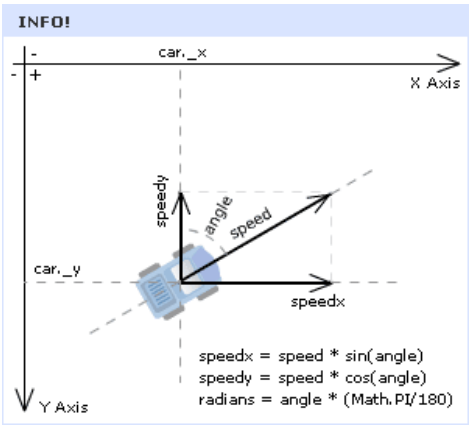
Funkcijo step() najdemo v Layer-ju Actions. Obravnavali jo bomo po kosih. Koda je lepo pokomentirana in zato njeno razumevanje ni težavno.

```
function step(who) {
    //check to see if the car in question is controlled by the player or by the computer
    if (_root["car"+who].code == "player") {
        //we will constantly decrease speed by multiplying it with a number below 1
        if (this["speed"+who]>0.3) {
1           this["speed"+who] *= _root.speedDecay;
        } else {
            this["speed"+who] = 0;
        }
        //the car will react to certain keys
        //accelerate
        if (Key.isDown(Key.UP) && this["speed"+who]<_root.maxSpeed) {
2           this["speed"+who] += _root.acceleration;
        }
        //brake (reverse)
        if (Key.isDown(Key.DOWN)) {
3           this["speed"+who] -= _root.backSpeed;
        }
        //steer left
        if (Key.isDown(Key.LEFT) && Math.abs(this["speed"+who])>0.3) {
4           _root["car"+who]._rotation -= _root.rotationStep*(this["speed"+who]/_root.maxSpeed);
        }
        //steer right
        if (Key.isDown(Key.RIGHT) && Math.abs(this["speed"+who])>0.3) {
5           _root["car"+who]._rotation += _root.rotationStep*(this["speed"+who]/_root.maxSpeed);
        }
        this["rotation"+who] = _root["car"+who]._rotation;
        //we calculate the two components of speed (X axis and Y axis)
        this["speedx"+who] = Math.sin(this["rotation"+who]*(Math.PI/180))*this["speed"+who];
6       this["speedy"+who] = Math.cos(this["rotation"+who]*(Math.PI/180))*this["speed"+who]*-1;
        //apply the components on the actual position of the car
        _root["car"+who]._x += this["speedx"+who];
        _root["car"+who]._y += this["speedy"+who];
```

Zgornja koda se nanaša na upravljanje avta:

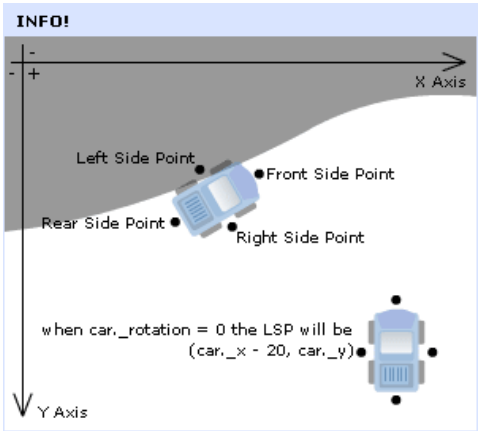
1. Če avto kar pustimo, bo upočasnjeval. Ko pa gre počasneje kot 0,3 slikovne elemente v Frame-u, ga kar ustavimo.

2. Če uporabnik stisne smerno tipko GOR in avto še ni dosegel maksimalne hitrosti, povečamo hitrost avta za vrednost pospeška (iz Layerja Defs).
3. Isto za smerno tipko DOL.
4. Če se avto še premika in uporabnik stisne smerno tipko LEVO, se bo avto zarotiral v levo za vrednost, ki je odvisna od koraka rotacije (rotationStep iz Layerja Defs) in od hitrosti avta.
5. Isto za smerno tipko DESNO.
6. Preostali del kode preračnunava X in Y komponento hitrosti. Ko se avto ne giblje direktno v smeri osi X ali Y, moramo izračunati njegove komponente hitrosti v smeri X in v smeri Y. Ilustracija je na spodnji sliki.



Preostali del kode je namenjen zaznavanju trkov med avtom in zelenico.

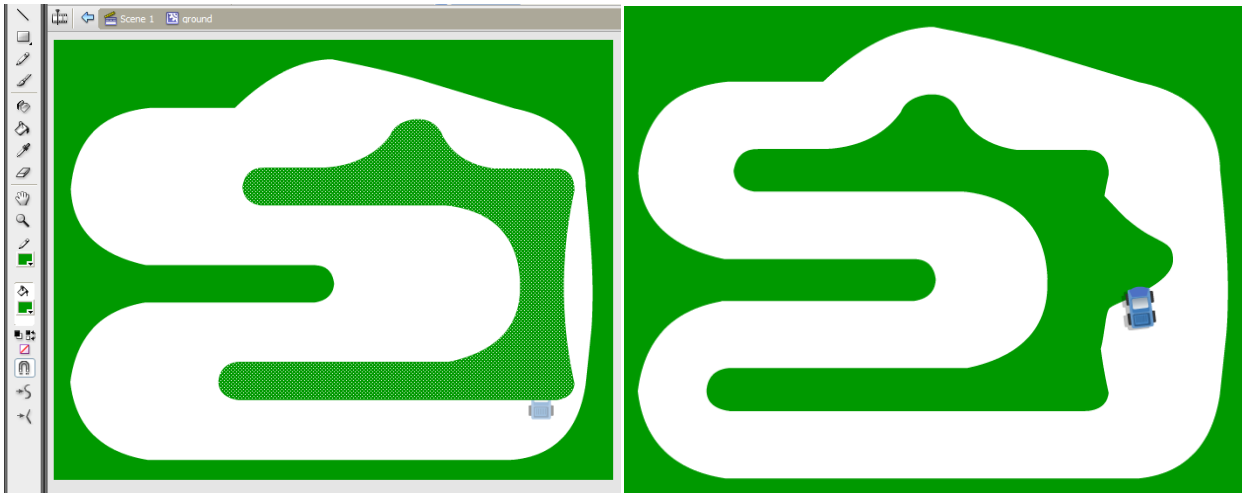
1. Določimo štiri kontrolne točke za zaznavanje kontakta, tako kot je razloženo na sliki.
2. Definiramo krajša imena za spremenljivke točk.
3. Če je leva točka na zelenici, zarotiramo avto in mu zmanjšamo hitrost.
4. - 6. Isto za preostale točke
7. Avto ima tudi senco, ki ji spreminjamo položaj glede na položaj avta.



```
35          //the collisions
36          //define the four collision points
37          _root["car"+who].pointLeft = {x:-20, y:0};
38          _root["car"+who].localToGlobal(_root["car"+who].pointLeft);
39          _root["car"+who].pointRight = {x:20, y:0};
40      1   _root["car"+who].localToGlobal(_root["car"+who].pointRight);
41          _root["car"+who].pointFront = {x:0, y:-25};
42          _root["car"+who].localToGlobal(_root["car"+who].pointFront);
43          _root["car"+who].pointBack = {x:0, y:25};
44          _root["car"+who].localToGlobal(_root["car"+who].pointBack);
45          //let's use some shorter variable names :)
46          this["lpx"+who] = _root["car"+who].pointLeft.x;
47          this["lpy"+who] = _root["car"+who].pointLeft.y;
48          this["rpx"+who] = _root["car"+who].pointRight.x;
49      2   this["rpy"+who] = _root["car"+who].pointRight.y;
50          this["fpx"+who] = _root["car"+who].pointFront.x;
51          this["fpy"+who] = _root["car"+who].pointFront.y;
52          this["bpx"+who] = _root["car"+who].pointBack.x;
53          this["bpy"+who] = _root["car"+who].pointBack.y;
54
55          //check for collisions
56          if (_root.terrain.hitTest(this["lpx"+who], this["lpy"+who], true)){
57      3       _root["car"+who]._rotation += 5;
58              this["speed"+who] *= 0.85;
59          }
60          if (_root.terrain.hitTest(this["rpx"+who], this["rpy"+who], true)){
61      4       _root["car"+who]._rotation -= 5;
62              this["speed"+who] *= 0.85;
63          }
64          if (_root.terrain.hitTest(this["fpx"+who], this["fpy"+who], true)){
65      5       this["speed"+who] = -1;
66          }
67          if (_root.terrain.hitTest(this["bpx"+who], this["bpy"+who], true)){
68      6       this["speed"+who] = 1;
69          }
70
71          //position the shadow of the car
72          _root["shadow"+who]._x = _root["car"+who]._x-4;
73      7   _root["shadow"+who]._y = _root["car"+who]._y+2;
74          _root["shadow"+who]._rotation = _root["car"+who]._rotation;
75      }
76      if (_root["car"+who].code == "computer") {
77      }
78  }
```

Spreminjanje ozadja

Najprej si poglejmo kako je narejeno ozadje, ki je trenutno v uporabi. Če kliknemo dvakrat na ozadje, ki je tudi iz razreda MovieClip, se bomo znašli v njegovem Timeline-u in bomo lahko manipulirali z njegovimi grafičnimi elementi**. Če s kurzorjem primemo ozadje in ga povlečemo, bomo opazili, da je pravzaprav sestavljeno iz dveh kosov - notranjega in zunanjega (spodnja slika levo).** Proga po kateri se giblje avto je praznina med dvema kosoma ozadja. To pravzaprav omogoča zaznavanje kontakta med avtom in zelenico.



Če sedaj "dorišemo" svoj del ozadja, gremo nazaj v glavni Timeline in testiramo igrico, bomo videli, da igra zazna tudi kontakt med avtom in novim delom zelenice (zgornja slika desno).

Torej, če hočemo zaznavati kontakt med avtom in zelenico, avto moramo premikati po "prazni" progi, kar pa je nemogoče v primeru, da kot progo vzamemo kar sliko. To bomo rešili tako, da bomo uvozili sliko, čez njo narisali progo tako kot zgoraj, in nato sliko proge premaknili v nov Layer, ki ne bo vplival na zaznavanje kontakta.

Postopek poteka tako:

1. Izbrišemo ozadje iz Stage-a in tudi iz knjižnice (Library)
2. Kliknemo na Layer **ground** in uvozimo sliko novega ozadja (File -> Import -> Import to stage)
3. Sliko ozadja prilagodimo velikosti Stage-a in jo nato pretvorimo v Movie clip (Modify -> Convert to symbol -> MovieClip). MovieClip poimenujemo kar **ground**. Za njegov instanceName vpišemo **terrain.**
4. V glavnem Timeline-u dvakrat kliknemo na novo ozadje, tako da se nam odpre njegov Timeline. Tu dodamo še en Layer, ki ga postavimo nad obstoječi Layer s sliko ozadja.
5. V novem Layerju z orodjem "Pen tool" naredimo oris zunanjega roba celotne slike in oris zunanjega ter notranjega roba same proge. Rišemo kar čez sliko proge. Če izklopimo vidljivost Layerja s progo (ikona očesa), bomo videli samo oris.
6. Z orodjem za selekcijo (zgornja, črna puščica) izberemo vse orise in gremo na menu Modify -> Combine objects -> Union. Na ta način bomo združili vse orise.

7. Sedaj lahko pobarvamo zelenico med orisi. Ko končamo, izbrišemo Layer z grafiko ozadja in gremo nazaj na glavni Timeline.
8. V njem naredimo nov Layer in ga poimenujemo GrafikaOzadja. Ko smo ga naredili, ga izberemo in nato v njega povlečemo sliko novega ozadja iz knjižnice (track1.jpg).
9. Stisnemo CTRL + ENTER in testiramo igro. Avto po potrebi lahko tudi pomanjšamo in ga prestavimo na štartno črto.