Stichting NIOC en de NIOC kennisbank

Stichting NIOC (www.nioc.nl) stelt zich conform zijn statuten tot doel: het realiseren van congressen over informatica onderwijs en voorts al hetgeen met een en ander rechtstreeks of zijdelings verband houdt of daartoe bevorderlijk kan zijn, alles in de ruimste zin des woords.

De stichting NIOC neemt de archivering van de resultaten van de congressen voor zijn rekening. De website www.nioc.nl ontsluit onder "Eerdere congressen" de gearchiveerde websites van eerdere congressen. De vele afzonderlijke congresbijdragen zijn opgenomen in een kennisbank die via dezelfde website onder "NIOC kennisbank" ontsloten wordt.

Op dit moment bevat de NIOC kennisbank alle bijdragen, incl. die van het laatste congres (NIOC2023, gehouden op donderdag 30 maart 2023 jl. en georganiseerd door NHL Stenden Hogeschool). Bij elkaar bijna 1500 bijdragen!

We roepen je op, na het lezen van het document dat door jou is gedownload, de auteur(s) feedback te geven. Dit kan door je te registreren als gebruiker van de NIOC kennisbank. Na registratie krijg je bericht hoe in te loggen op de NIOC kennisbank.

Het eerstvolgende NIOC vindt plaats op donderdag 27 maart 2025 in Zwolle en wordt dan georganiseerd door Hogeschool Windesheim. Kijk op www.nioc2025.nl voor meer informatie.

Wil je op de hoogte blijven van de ontwikkeling rond Stichting NIOC en de NIOC kennisbank, schrijf je dan in op de nieuwsbrief via

www.nioc.nl/nioc-kennisbank/aanmelden nieuwsbrief

Reacties over de NIOC kennisbank en de inhoud daarvan kun je richten aan de beheerder:

R. Smedinga kennisbank@nioc.nl.

Vermeld bij reacties jouw naam en telefoonnummer voor nader contact.

19

Het Soccer-Funproject: functioneel programmeren in het onderwijs met behulp van voetbal

Peter Achten - Model Based System Development, Informatica, Radboud Universiteit Nijmegen
P.Achten@cs.ru.nl

SAMENVATTING

In het Soccer-Funproject leren scholieren en studenten op een stimulerende manier functionele programmeertaalconcepten. Soccer-Fun is geïnspireerd op een uitspraak van Johan Cruijff, waarin hij stelt dat het brein van een voetballer een zuivere functie is. In Soccer-Fun maken scholieren en studenten een breinfunctie en proberen daarmee andere teams te verslaan.

TREFWOORDEN

Functioneel programmeren, strategieën, simulatie, kunstmatige intelligentie

INLEIDING

Het Soccer-Funproject is ontwikkeld aan de Radboud Universiteit Nijmegen om informaticastudenten op een stimulerende manier te laten oefenen met functioneel programmeren en redeneren over software. Hierbij gaat het om software die bestaat uit verschillende componenten die tegelijkertijd beslissingen nemen. Soccer-Fun is geïnspireerd op de volgende uitspraak van Johan Cruijff:

'Als ik een bal aan de voet heb die ik wil afspelen, dan moet ik rekening houden met mijn bewaker, de wind, het gras, de snelheid waarmee de spelers lopen. Wij berekenen de kracht waarmee je moet schoppen en de richting waarin in ééntiende seconde. De computer doet daar twee minuten over!' (De Tijd, 2 mei 1987)

Hiermee zegt Johan Cruijff in feite dat zijn brein een functie is die een aantal parameters krijgt (bewaker, wind, gras, loopsnelheden) en die twee waarden berekent (kracht en richting). Dit karakteriseert meteen het doel van de opdrachten waarmee studenten zich bezighouden met Soccer-Fun: maak je eigen breinfunctie, stop die in een elftal van voetballers, en kijk of je andere teams kunt verslaan.

SOCCER-FUN

In het bachelorinformaticacurriculum aan de Radboud Universiteit Nijmegen worden de *imperatieve, objectgeoriënteerde* en *functionele* programmeerparadigma's onderwezen. Het functionele programmeerparadigma stimuleert het modelleren van probleemdomeinen in termen van data types, abstractie en compositie in programmeren met behulp van *(hogere-orde)functies*, en het bewerken van gegevensverzamelingen met *lijsten*. Functionele talen zijn uitermate geschikt voor abstractie en dat komt sterk tot uiting in de bijbehorende practicumopdrachten. Dit geldt ook voor het door ons ontwikkelde practicum voor de tweedejaarscursus *'Abstractie en Compositie in Programmeren'* (2005-2008)[4]. De abstracte aard van de practicumopdrachten wordt echter niet door alle studenten gewaardeerd. Om een (nieuw) programmeerparadigma onder de knie te krijgen is actieve deelname aan het practicum een bijna noodzakelijke voorwaarde. Het is dus belangrijk om een practicum aan te bieden dat studenten stimuleert hun taalvaardigheid en probleemoplossend vermogen te vergroten. Bovendien willen wij als docenten graag de *fun* van functioneel programmeren overbrengen op onze studenten. Het Soccer-Fun-raamwerk is ontwikkeld met deze twee doelen voor ogen. Voor de volledigheid willen we benadrukken dat het Soccer-Funraamwerk een *aanvullende* rol heeft in het pakket practicumopdrachten dat we studenten aanbieden.

Soccer-Fun is geïnspireerd op de hierboven geciteerde uitspraak van Johan Cruijff waarin hij stelt dat het brein van een voetballer een functie is. In het citaat wordt alleen de situatie beschreven waarin een voetballer in balbezit is en de bal wil spelen. In een echte wedstrijd treden meer situaties op zoals in balbezit komen, (strategische) posities innemen, verdedigen en aanvallen. Merk op dat dit niet alleen in voetbal voorkomt, maar in iedere teambalsport zoals hockey, basketbal en handbal. Dit draagt ertoe bij dat het ontwikkelen van breinfuncties ook interessant is voor studenten die geen affiniteit hebben met voetbal. Uiteraard moeten studenten rekening houden met specifieke voetbalsituaties zoals de buitenspelregel. Het bedenken en implementeren van succesvolle strategieën is uitdagend voor de informatica- en kunstmatigeintelligentiestudenten. Studenten kunnen de kwaliteit van hun implementatie toetsen door de door hen ontwikkelde teams tegen elkaar te laten spelen, zoals getoond wordt in figuur 1. *Soccer-Fun* voegt op deze manier een *competitie-element* toe aan het practicum, wat als stimulerend ervaren wordt. Dit is niet vrijblijvend voor de studenten: de cursus wordt afgesloten met een toernooi waarin alle teams die voldoen aan vooropgestelde criteria tegen elkaar uitkomen. We loven twee prijzen uit: één voor de toernooiwinnaar en één voor de code die in onze ogen het best gebruik maakt van functionele programmeertaalconcepten.

Figuur 1:
Het Soccer-Funraamwerk in actie.

[4] *Inmiddels is deze cursus opgevolgd door "Functioneel Programmeren" voor de studies informatica en kunstmatige intelligentie.*

Soccer-Fun is geïmplementeerd in de in Nijmegen ontwikkelde zuivere functionele programmeertaal Clean. Deze is gratis beschikbaar en kan gedownload worden via http://clean.cs.ru.nl/.
Het *Soccer-Funproject* zelf is een open project, d.w.z. dat iedereen, na aanmelding, bijdragen kan leveren in de vorm van code, oefeningen, voorbeeldteams, documentatie, en dergelijke. De *Soccer-Funsite* is: http://www.cs.ru.nl/P.Achten/SoccerFun/SoccerFun.html
Om met *Soccer-Fun* te kunnen werken moeten studenten kennisgemaakt hebben met Clean taalelementen die ook in andere programmeertalen te vinden zijn zoals basistypes en records. Kenmerkende functionele taalelementen die ook nodig zijn, zijn algebraïsche types, lijsten en lijstcomprehensions. In eerste instantie kunnen taalconcepten als hogere-ordefuncties en recursief programmeren 'vermeden' worden. Deze komen aan bod naarmate de breinfuncties uitdagender worden. Op deze wijze kan functioneel programmeren stapsgewijs worden aangeboden en geïntegreerd in een bestaand of een te ontwikkelen programmeerpracticum.

Het voetbalbrein

In zijn eenvoudigste vorm is het brein van de voetballer een functie (*FootballerAI'*) van brein invoer (*BrainInput*) naar brein uitvoer (*BrainOutput*):
:: FootballerAI' :== BrainInput → BrainOutput
BrainInput is een record waarin de breinfunctie de beslissingen van de scheidsrechter aantreft, de toestand van de voetbal (positie en snelheid), alle overige spelers in het veld en de speler zelf. Het doel van de breinfunctie is een passende actie te berekenen van type *BrainOutput*. De belangrijkste acties zijn het bewegen van de speler, het spelen van de bal met been of hoofd en het in bezit nemen van de bal. Het Soccer-Funraamwerk interpreteert de gewenste acties en zet deze om in realistische acties. In het Soccer-Funraamwerk kan de student er voor kiezen of er middels randomwaarden afwijkingen toegevoegd worden aan deze acties.

Zodra het voetbalbrein een echte wedstrijd moet spelen, is het van belang dat deze naar de scheidsrechter luistert. Deze geeft namelijk aan wanneer de eerste helft van de wedstrijd voorbij is, en er dus een helftwissel plaatsvindt. Om dit te realiseren heeft het brein een *geheugen* nodig. Dit is de normale vorm van de breinfunctie (*FootballerAI*):
:: FootballerAI memory :== (BrainInput, memory) → (BrainOutput, memory)
Dit is een voorbeeld van een geparametriseerd functie type, waarin memory het type geheugen is van de voetballer. De student ontwerpt een geschikt geheugen (meestal een record). Dit geheugen wordt als argument meegegeven bij aanroep van de breinfunctie, die naast een voetballeractie van type *BrainOutput* ook een nieuwe waarde van type *memory* oplevert.

Breinschets

In het informaticaprogrammeeronderwijs vinden we het belangrijk dat studenten een ontwerp maken voordat de codeerfase begint. In *Soccer-Fun* kun je al op een informele, maar toch effectieve, manier nadenken over je breinfunctie. We noteren dat met *breinschets*. Hier is een voorbeeld dat het spelgedrag van mini-F-voetballertjes[5] beschrijft:

mini-effie voetbalveld doel invoer
if ik kan tegen de bal schoppen
==> schop bal richting doel tegenstander
otherwise ==> ren en draai richting bal

[5] *Kinderen die graag willen voetballen, maar nog geen 7 jaar zijn kunnen als 'mini-F' lid worden van een voetbalvereniging en meedoen aan competitie.*

Een breinschets heeft een *naam (mini-effie)* en als argument ten minste de *BrainInput (invoer)*. De uitvoer is ten minste een *BrainOutput*-waarde (*schop, ren, draai*). Met behulp van beslissingsregels wordt de werking van het brein vastgelegd. Op deze manier wordt het voor de studenten snel duidelijk of ze voldoende informatie kunnen vinden in de BrainInput, of dat ze extra informatie nodig hebben. In dit geval zijn dat de afmetingen van het voetbalveld en de positie van het te verdedigen doel. Deze twee argumenten krijgt een voetballer mee aan het begin van een wedstrijd. Alle benodigde extra informatie wordt als argument meegegeven aan de breinschets vóór het verplichte invoerargument[6]. Een breinfunctie die geen geheugen nodig heeft hoeft dit ook niet in zijn breinschets op te nemen, en kan als een *FootballerAI'* functie gerealiseerd worden. Een breinfunctie die wel een geheugen nodig heeft, neemt dit bij zijn invoer argument op en levert dit ook op als resultaat. Een dergelijk brein is uiteraard van type *(FootballerAI memory)*.

Compositioneel programmeren is een belangrijke functionele programmeerstijl die ook toegepast kan worden in *Soccer-Fun*. Hierin worden breinfuncties samengesteld middels eenvoudigere breinfuncties. Als voorbeeld geven we de breinschetsen van drie elementaire breinfuncties: *halt, fix en kick*.

halt invoer
==> *beweeg niet en draai niet*

fix positie d invoer
if de afstand tussen mij en positie is minder dan *d* meter
==> *halt invoer*
otherwise ==> *ren en draai richting positie*

[6] *De technische reden hiervoor is dat in functionele talen currying het mogelijk maakt om extra informatie toe te voegen aan een functie vóór de 'normale' argumenten.*

kick positie invoer
if ik kan tegen de bal schoppen
==> *schop bal richting positie*
otherwise ==> *halt invoer*

Het *halt*-brein besluit onder alle omstandigheden stil te staan en niet te roteren. Het *(fix p d)*-brein beweegt een speler naar positie *p* op het voetbalveld en staat stil zodra de speler positie *p* op afstand *d* genaderd is. Het *(kick p)*-brein laat de speler de bal naar positie *p* schoppen indien de bal binnen trapbereik ligt.
Met deze elementaire breinfuncties kan het *mini-effie*-brein eenvoudig gerealiseerd worden:

mini-effie voetbalveld doel invoer
if ik kan tegen de bal schoppen
==> *kick* (centrum doel tegenstander)
invoer
otherwise ==> *fix* balpositie nul *invoer*

Het gebruik van breinschetsen is een nuttig hulpmiddel voor docent en student om ideeën over mogelijke breinfuncties te communiceren, bekritiseren, en verder te ontwikkelen. Door het kiezen van de mate van detail kan een concrete programmeeropdracht gegeven worden en wordt tegelijkertijd veel keuzevrijheid geboden om deze te realiseren.

Scheidsrechterbeslissingen

In *Soccer-Fun* is een scheidsrechter geïmplementeerd die de concrete speelacties van alle spelers beoordeelt, de spelregels hanteert, en op grond hiervan een reeks *scheidsrechterbeslissingen* genereert. Deze beslissingen worden aan alle spelers bekend gemaakt middels hun *BrainInput*. Net als het brein van de voetballers kan het brein van een scheidsrechter met een functie gemodelleerd worden. In tegenstelling tot voetballers die een actie berekenen, is het doel van de scheidsrechter het *monitoren* van de wedstrijd en indien nodig spelers te sanctioneren. De scheidsrechter

kan dus ook ingezet worden om voetballers bepaalde trainingen te laten doen. In *Soccer-Fun* is een aantal scheidsrechters ontwikkeld die de voortgang van de volgende trainingen in de gaten houden en van commentaar voorzien:

- *Slalomtraining:* implementeer een voetbalbrein dat een voetballer van de ene kant van het voetbalveld naar de andere kant laat lopen zodanig dat deze om de achtereenvolgende tegenstanders slalomt. Aan de andere kant ligt de voetbal klaar om in de goal gespeeld te worden om de opdracht succesvol af te ronden.
- *Overspeeltraining:* implementeer een voetbalbrein dat een voetballer de bal laat spelen naar de eerstvolgende teamgenoot die dichter bij het doel van de tegenstander staat. De teamgenoot mag slechts een beperkt aantal meter naar de bal toelopen om deze in bezit te nemen. De laatste voetballer moet de bal in de goal spelen om de opdracht succesvol te beëindigen.
- *Vrij-overspelentraining:* implementeer een voetbalbrein dat in staat is het goede moment te kiezen om de bal te spelen naar een medespeler zodanig dat geen tegenstander in staat is de bal te onderscheppen. De medespeler moet in balbezit komen zonder ver te lopen om de opdracht succesvol te beëindigen.
- *Keepertraining:* implementeer het voetbalbrein van een keeper die het midden van zijn goal verdedigt tegen een aantal tegenstanders die de bal naar elkaar overspelen. De keeper moet dit een bepaalde periode lang doen om te slagen voor deze opdracht.

De hierboven beschreven trainingsopdrachten lenen zich goed voor het werken met lijsten (de verzameling van spelers op het veld) en lijstcomprehensions. Voorbeelden van functies die met lijsten werken zijn: bepaal wie je teamgenoten zijn, waar deze staan, welke teamgenoot het dichtst bij je staat, welke teamgenoten niet aanspeelbaar zijn omdat ze zich in buitenspelpositie bevinden of omringd zijn door te veel tegenstanders. Dit bereidt studenten tevens voor op het maken van breinfuncties die een echte wedstrijd gaan spelen.

De effectiviteit van een team wordt voor een groot deel bepaald door de geïmplementeerde strategie. Een team dat de hierboven beschreven strategie van de mini-F-spelers hanteert is normaal gesproken eenvoudig te verslaan door het innemen van veldposities en het overspelen van de bal. De student moet dus nadenken over een geschikte strategie en deze implementeren. Voorbeelden van functies die in dit kader ontwikkeld kunnen worden zijn: hoe zorg ik ervoor dat mijn team in balbezit komt? wat is een geschikte veldverdeling voor mijn spelers? wat is een effectieve aanval? wat is een effectieve verdediging? In de cursus ontwikkelen studenten een voetbalteam waarin de spelers uitgerust worden met de door hen ontwikkelde voetbalbreinen. Dit voetbalteam komt uiteindelijk in actie in een toernooi dat de cursus afsluit. Om hiervoor in aanmerking te komen moet het team aan een aantal kwaliteitseisen voldoen:

- De keeper mag het strafschopgebied niet verlaten (hij mag zich niet als een *mini-F-voetballer* gedragen) en moet de bal onderscheppen als dat redelijkerwijs mogelijk is (hij moet dus actief deelnemen aan het spel).
- Veldspelers moeten een redelijke veldverdeling innemen (om te voorkomen dat ze allemaal achter de voetbal aanhollen). Veldspelers moeten de bal afspelen als dat redelijkerwijs zinvol is (om te voorkomen dat men een team van solisten ontwerpt).
- Alle voetballers dienen de spelregels in acht te nemen en de beslissingen van de scheidsrechter op te volgen. Dit is vooral bedoeld om te voorkomen dat een spel in livelock komt omdat voortdurend het verkeerde team de bal wil inwerpen of aftrappen en de scheidsrechter hierop moet ingrijpen.
- De breinfuncties moeten voldoende efficiënt zijn. Dit kan eenvoudig bepaald worden door een team tegen zichzelf te laten spelen en te eisen dat de framerate niet lager wordt dan 20 frames per seconde. In de Soccer-Fun-GUI wordt deze teller continu getoond.

Soccer-Fun voor vwo'ers

Naast de inzet in het tot nu toe beschreven academisch onderwijs, is Soccer-Fun ook gebruikt tijdens wervingsactiviteiten voor vwo-scholieren om informatica te gaan studeren aan de Radboud Universiteit Nijmegen. Uiteraard ontbreekt bij deze voorlichtingsactiviteiten de tijd en voorkennis bij de deelnemers om geheel zelfstandig breinfuncties te ontwikkelen. Om hen hiermee te helpen krijgen ze de beschikking over een aantal voorgemaakte breinfuncties waaraan een aantal kleine aanpassingen gemaakt moeten worden. Het gebruik van breinschetsen blijkt een goed hulpmiddel te zijn om vwo-scholieren uit te leggen hoe deze breinfuncties werken, en hoe ze de gewenste verandering kunnen implementeren. Vwo'ers reageren enthousiast op Soccer-Fun, en zijn veelal in staat om op het niveau van breinschetsen strategieën te begrijpen en aan te passen. Voor een daadwerkelijke realisatie in Clean ontbreekt vaak de tijd. We willen onderzoeken op welke manier het abstractieniveau van de huidige Soccer-Fun-API verhoogd kan worden.

CONCLUSIE

We stellen vast dat *Soccer-Fun* een welkome aanvulling is op bestaande practica functioneel programmeren. Studenten leren functionele taalconcepten toepassen zoals algebraïsche types, (hogere-orde)functies, en lijst(comprehensions) die normaal gesproken als abstract ervaren worden, maar die geconcretiseerd worden binnen Soccer-Fun. De meerderheid van de studenten reageert enthousiast op Soccer-Fun. Vanwege onze positieve ervaringen bij het inzetten van Soccer-Fun in voorlichtingsactiviteiten voor vwo-scholieren, willen we onderzoeken of het mogelijk is om onderwijsmateriaal te ontwikkelen voor het vwo om leerlingen met functionele taalconcepten te laten werken.

VERANTWOORDING

In het verleden hebben we ervaring opgedaan in het verzorgen van voorlichtingsactiviteiten met behulp van Mathew Nelsons Robocodeproject (http://robocode.sourceforge.net/). Dit is een simulatiepakket om het objectgeoriënteerde paradigma uit te leggen en om ermee te werken. Sleutelelementen die we hebben overgenomen in *Soccer-Fun* zijn de grafische visualisatie van de voetballers en het competitie-element (in *Robocode* strijden tanks tegen elkaar). Om het abstractieniveau van *Soccer-Fun* te verhogen willen we onderzoeken in hoeverre de methoden in het objectgeoriënteerde Alice-raamwerk [Conway 2000] van toepassing zijn op *Soccer-Fun*. In Alice kunnen gebruikers complexe acties ontwikkelen in een combinatorische stijl die erg lijkt op de functionele stijl. Een ander sleutelelement van Alice is dat code tijdens de ontwikkelfase direct geëxecuteerd kan worden. Op deze manier kunnen vooral beginnende gebruikers direct het effect van hun code zien. In het simulatiepakket NetLogo [Wilensky 1999] wordt door middel van scripts die op turtles toegepast worden complexe, dynamische, systemen beschreven. Soccer-Fun onderscheidt zich hiervan door het brein als een pure functie te zien, en het gebruik van functionele taalconcepten om de breinfunctie te implementeren.

↑

[1]
Een referencemanual is te vinden in de Soccer-Fun distributie.
[2]
Peter Achten. Teaching Functional Programming with Soccer-Fun. In Frank Huch, Adam Parkin, Eds. Proceedings of the 2008 ACM SIGPLAN Workshop on Functional and Declarative Programming in Education (FDPE'08), Victoria (BC) Canada, September 21, 2008, ISBN:978-1-60558-068-5, pp. 61-72, 2008.
[3]
Matthew Conway et al. Alice: Lessons from building a 3D system for novices. In Thea Turner, Gerd Szwillus, Eds. CHI'00: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, New York, NY, USA, ACM, pp. 486-493, 2000.
[4]
Uri Wilensky. NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL, 1999.