VRIJE UNIVERSITEIT BRUSSEL

Project Computersystemen: Vier op een rij

Rodolfo Perez Tobar

December 26, 2021

Academiejaar 2021-2022

Bachelor Computerwetenschappen

Inhoudstafel

1 Inleiding

1.1 Intro

Voor het vak computersystemen kregen we als project om een "eenvoudig" spel te coderen in assembly. Ik heb gekozen voor vier op een rij. Een spel dat bijna iedereen kent. De spelregels zijn redelijk eenvoudig en gemakkelijk te begrijpen. Na de goedkeuring van het project begon ik eraan te werken. Vooraleer we de procedures gaan overlopen eerst nog een kleine herinnering van de spelregels.

1.2 Spelregels

Vier op een rij is een spel dat je met twee speelt. In dit geval speelt de speler tegen een andere speler. Elke speler krijgt een kleur toegekend meestal is het geel of rood, maar in principe is eender welke kleur goed. Men speelt in een rooster waar het enkel mogelijk is om een munt toe te voegen vanuit helemaal van boven. Elke speler speelt om de beurt. Het doel van het spel is om als eerste vier munten van uw kleur naast elkaar te krijgen dit kan zowel horizontaal, verticaal als diagonaal en tegelijkertijd proberen te voorkomen dat jouw tegenstander vier munten van zijn kleur naast elkaar kan krijgen.

2 Handleiding

Hier zullen we snel uitleggen welke toets gebruiken om door het menu te navigeren, het spel opstarten en spelen. In elke menu staat er duidelijk welke toets je moet indrukken voor een bepaalde actie uit te voeren. Maar we nemen snel een kijkje naar de functionaliteiten.
In het startmenu kan je elke de volgende acties uitvoeren: een nieuw spel starten door op "spatie" te drukken, een kijkje nemen naar de stand van het spel (hoeveel keer heeft elke speler al gewonnen) door op "s" te drukken, een verfrissing van de spelregels bekijken door op "r" te drukken en de applicatie sluiten door op "esc" te drukken.
In elk van de submenu's (de regels herbekijken of de standen bekijken) kan je enkel op "b" drukken om terug te komen naar het start menu. De standen die in statistics menu worden bijgehouden zijn enkel geldig voor de huidige executie van de applicatie, na het afsluiten van de applicatie door op de start menu op "esc" te drukken zullen de standen worden herstart op 0.
Als je op "spatie" had gedrukt dan zit ga je een nieuw spel starten, maar vooraleer je dit doet kan je op "b" drukken om terug te komen naar het start menu of een keuze maken van het gewenste spelbord door op een getal te drukken tussen "0" en "9" afhankelijk van het bord grootte zal dit variëren.
Nadat de selectie van het speelbord is gebeurd kan je op "1" of "2" te drukken op te kiezen welke speler gaat beginnen of als je een foute grootte hebt gekozen terugkeren naar het vorige menu door op "b" te drukken.
Tijdens het spel kan je het spel pauzeren door op "p" te drukken. Ook is het mogelijk om de laatste beweging die gemaakt werd ongedaan maken(undo) door op "d" te drukken.
Let op, je moet wel tweemaal op de num-lock toets drukken om de numberpad te activeren. Het spel is zodanig geïmplementeerd dat de personen die geen numeriek toetsenbord hebben op hun pc de kolom kunnen kiezen waar ze hun munt willen plaatsen door op de de shift-toets te drukken en nul tot en met negen van de cijfers bovenaan het toetsenbord.
Na afloop van een potje heb je de keuze om verschillende acties te doen. Je kan op "e" drukken om een nieuw potje te lanceren, waarbij je weer zal moeten kiezen wie er als eerste gaat. Je kan terug naar het begin menu gaan door op "m" te drukken. Je kan kijken naar de standen van het spel door op "s" te drukken of je kan de applicatie verlaten door op "esc" te drukken.

3 Programma

We gaan het nu hebben over de code. We gaan de verschillende procedures kort overlopen en uitleggen wat ze doen. De code bestaat uit 3 grote onderdelen: de globale omgeving, het codesegment en het datasegment. We zullen eerst kijken naar de globale omgeving vervolgens naar het codesegment en als laatste het datasegment.

3.1 Globale omgeving

In het begin van de code worden globale constanten toegekend.

3.1.1 Constanten

De constanten die hier worden gebruikt, zijn nodig om de videomodus te kunnen bedienen. Deze constanten zijn: VMEMADR, SCRWIDTH en SCRHEIGHT.

- VMEMADR is om aan te geven waar de video memory begint.
- SCRWIDTH is de breedte van het scherm.
- SCRHEIGHT is de hoogte van het scherm.

3.2 Codesegment

Het volgende onderdeel is het CODESEG in dit gedeelte komen alle procedures aan bod die het spel tot stand brengen. Hierin is er ook nog een kleine verdeling, namelijk alle globale procedures die ervoor zorgen dat de Gui goed werkt (global procedures), de procedures die array's gaan manipuleren(array procedures), de procedures die de graphics helpen voor te stellen(graphics), wat nodig is om de menus juiste op het scherm te tekenen (menus), de print procedures die ervoor zorgen dat je tekst kan tonen op het scherm(printing), de spellogica(gamelogic) en de gameloop zelf.

3.2.1 Global procedures

Hierin kan je de procedures setVideoMode, terminateProcess en moveCursor terugvinden.

- setVideoMode is nodig om de overgang van textmode(03h) naar videomode(13h) te kunnen doen.
- terminateProcess zorgt ervoor dat je de videomode goed afsluit en zo terugkeert naar de textmode. Hierin zal ook een afscheid bericht op het scherm worden getoond om de speler te bedanken voor het gebruik van de applicatie.
- moveCursor zorgt ervoor dat je de textcursuror op je scherm kan verplaatsten om zo tekst te schrijven op het scherm.

3.2.2 Printing

Hierin kan je de procedures printChar, printString en printScore terugvinden.

- printChar zorgt ervoor dat je één ascii character op het scherm kan laten verschijnen.
- printString maakt gebruik van printChar om zo zinnen te kunnen printen. Elke zin wordt steeds beïndigd door het speciale teken "$" Ook is er een teken om een newline aan te geven, deze is dit "~"speciale teken.
- printScore zorgt ervoor dat je de scores van de huidige executie kan printen in het menu Statistics.

3.2.3 Array procedures

Hierin kan je de procedures updateStatus, clearGrid, adaptGridvalues, adaptDrawGrid, adaptPieceDim, adaptFieldLogic, adaptCoordinates, adaptEnnumeration, adaptValidator, adaptWinCondition, adaptField en restoreField terugvinden.

- updateStatus zorgt ervoor dat je de array goed kan bijhouden om de correcete staat van het veld in de array te zetten.
- clearGrid zorgt ervoor dat je de array volledig kan resetten vooraleer je een nieuw spel gaat beginnen.
- restoreField zorgt ervoor dat je waaneer je het spel pauzeert en dan weer hervat dat het veld nog steeds correct wordt bijgewerkt.
- adaptGridvalues zorgt ervoor dat het aantal rijen en kolommen goed aangepast worden. Deze waarden zijn terug te vinden in gridValues.
- adaptDrawGrid zorgt ervoor dat de waarden die ervoor zorgen dat het speelveld goed getekend wordt goed aangepast worden. Deze waarde zijn terug te vinden in grid.
- adaptPieceDim zorgt ervoor dat de grootte van een stuk aanpast wordt afhankelijk van de grootte van het veld. Deze waarde is terug te vinden in pieceDim.
- adaptFieldLogic zorgt ervoor dat je de array met waarden van het spel goed kan interpreteren door de indicaties die het veld definiëren veranderen. Deze waarden zijn upperRightCorner, rowInBetween, firstTop en rowSeparation.
- adaptCoorinates zal ervoor zorgen dat de coördinaten van het veld op het scherm geüpdatet worden afhankelijk van het veld grootte. Deze waarden worden bijgehouden in vertical en horizontal. Je leest deze coördinaten in van links naar rechts. De waarden in vertical representeren de hokjes van onder naar boven. En de waarden in horizontal representeren de hokjes van links naar rechts.
- adaptEnnumeration zorgt ervoor dat de indicatie van de kolommen goed vertoond worden om de speler te kunnen informeren van wat elke nummertoets nu doet. De waarde die aangepast wordt bevindt zich in ennumeration.
- adaptValidator zorgt ervoor dat je de input values aanpast. Dus als je vijf kollommen hebt (0-4) en je druk op toets "5" dan mag er geen input geregistreerd worden. Met deze procedure kan je dit vermijden. De waarde die aangepast wordt bevind zich in validateInput.
- adaptWinCondition zorgt ervoor dat je de waardes die nodig zijn om de win conditie na te gaan kan aanpassen. Deze waardes bevinden zich in horCheck (voor de horizontale conditie), vertCheck (voor de verticale conditie), posCheck (voor de f(x)=x conditie) en negCheck (voor de f(x)=-x conditie).
- adaptField zorgt ervoor dat je alle adapt procedures in één procedre kan aanroepen om zo de code gemakkelijker te kunnen onderhouden.

3.2.4 Menus

Hierin kan je de procedures vinden die een menu op scherm correct laten verschijnen. Deze procedures zijn: menuDistribution, menuConfiguration en menuDisplay.

- menuDistribution zorgt ervoor dat de correcte header(title) op het scherm wordt vertoond.
- menuConfiguration zorgt ervoor dat de correcte buttons(interations die mogelijk zijn) op het scherm vertoond worden.
- menuDisplay combineert menuDistribution en menuConfiguration on zo de juiste menu te laten zien.

3.2.5 Graphics

Hierin kan je de procedures fillBackground, drawRectangle, drawGrid, drawMove, makeButton, playerTurn, announceInfo en changeTurn terugvinden.

- fillBackground zorgt ervoor dat je het volledige scherm kan vullen met 1 bepaalde kleur.
- drawRectangle zorgt ervoor dat je een rechthoek kan tekenen en sind een rechthoek een gearceerde rechthek kan zijn of enkel de omtrek moet je dit ook meegeven aan de procedure, of je enkel de omtrek wilt of een arcering.
- drawGrid zorgt ervoor dat je het speelveld kan tekenen, enkel het bord zonder de muntstukken.
- drawMove zal een muntstuk tekenen van de juiste kleur afhankelijk van de speler die aan de beurt is(geel is voor speler 1 een paars voor speler 2).
- makeButton zal ervoor zorgen dat je op het scherm een knop kan tekenen waarmee je kan interageren, daarvoor moet je de naam van de knop geven,de kleur van de tekst en zijn positie.
- changeTurn zal ervoor zorgen dat de beurt juist wordt veranderd op het scherm.
- playerTurn zorgt ervoor dat je de huidige speler aan de beurt is goed vertoond wordt. playerTurn wordt aangeroepen in changeTurn.
- announceInfo zorgt ervoor dat je na afloop van een spel de juiste winnaar kan aangeven als er iemand gewonnen heeft. Als er gellijk spel was dan geeft je dit ook aan.

3.2.6 Gamelogic

In dit onderdeel wordt de logica van het spel geïmplementeerd. Je zal hierin de volgende procedures tegenkomen: makeMove, fullCheck, winCondition, checkWinForDirection, checkWin en gameStatus.

- makeMove zorgt ervoor dat je een zet op het bord kan spelen.
- fullCheck zal gaan kijken of het bord volledig gelud is en dus geen enkel zet meer mogelijk is.
- winCondition zorgt ervoor dat we 1 bepaalde 4 op een rij kunnen na gaan.
- checkWinForDirection maakt gebruik van winCondition om alle mogelijke 4 op een rij na te gaan die mogelijk zijn in 1 bepaalde richting.
- checkWin maakt gebruik van checkWinForDirection om zo het heel veld af te gaan en op zoek te gaan naar één moglijke vier op een rij.
- gameStatus zorgt ervoor dat je gemakkelijk naar de status van het spel kan kijken(kan je verder spelen, heeft speler 1 gewonnen, heeft speler 2 gewonnen of is er gelijk spel). Deze states worden als volgt gerepersenteerd: 0 betekent speel verder,1 betekent speler 1 heeft gewonnen, 2 betekent speler 2 heeft gewonnen en 3 betekent er is gelijk spel.

3.2.7 Gameloop

In dit onderdeel vindt zicht het eigenlijke spel. Alle bovenstaande procedures (3.2.2 tot en met 3.2.6) worden hier gebruikt om zo het spel zelf te laten lopen. Hierin kom je 1 procedure tegen, namelijk: game.

- game is dus het eigenlijke spel, hierin wordt alles samengebracht om zo tot een werkend programma te komen.

3.2.8 Main

In dit onderdeel bevindt zicht het entry point voor het programma correct te laten verlopen, namelijk de procedure main.

- main heb je nodig om de executie goed te laten verlopen net zoals in alle C programma's, hierin wordt dus enkel game opgeroepen die het eigenlijke werk doet in de applicatie.

3.3 Datasegement

Als laaste onderdeel is het DATASEG in dit gedeelte komt alle data aan bod die nodig is om het spel te kunnen opstarten, manipuleren, afbeelden en spelen. Hier werden ook de elementen gegroepeerd per categorie. De categoriën zijn: constanten(constants), vectors used in adaptField, vectors gebruikt voor de menus, titels, extra tekst, interacties(interactions) en de spelregels(rules).

3.3.1 Titles

Hierin zitten de strings die de titels op het scherm zullen voorstellen bij elk menu. Onderandere vind je hier: connect4, titleRules, beginner, paused, winner, draw, turn, difficulty en statistics.

- connect4 is de titel in mainMenu.
- titleRules is de titel in rulesMenu.
- beginner is de titel in choiceMenu.
- paused is de titel in pauseMenu.
- winner is om aan te geven wie er heeft gewonnen in announceMenu.
- draw is om aan te geven dat er gelijk spel is in announceMenu.
- turn is om aan te geven wie aan de beurt is in gamemenu.
- statistics is de title in statisticsMenu.
- Difficulty is de titile je laat zien om de speler aan te geven dat hij een spelbord grootte moet kiezen.

3.3.2 Vectors menus

Hierin zitten de vectoren met offsets die gebruikt worden in een van de procedure menuDistribution en menuConfiguration afhankleijk van de vector alle vetoren behalve TextHeader worden in menuConfiguration gebruikt als 1 van de parameters.

- textHeader is de vector met alle titles om de juiste te kiezen afhankelijk van het menu.
- menuMain is de vector met alle elementen die nodig zijn om de main menu te tekenen.
- menuStats is de vector met alle elementen die nodig zijn om de statistics menu te tekenen.
- menuChoise is de vector met alle elementen die nodig zijn om de menu met de keuze van wie begint te tekenen.
- menuGame is de vector met alle elementen die nodig zijn om menu waar het spel gespeeld wordt te tekenen.
- menuAnnounce is de vector met alle elementen die nodig zijn om de announcement te doen aan het einde van een potje.
- menuDifficulty is de vector met alle elementen die nodig zijn om de menu waar je de grootte van het spelbord kiest te tekenen.

3.3.3 Extra text

Hierin zitten strings die geen titel is maar ook onderdelen kan zijn van de menus. Onderandere vind je hier: p1, p2, draws, credits, ennumeration en msg. Al deze elementen worden in de vectoren gestopt die ervoor zorgen dat menus goed getekend worden (zie 3.3.2). msg is de enige die geen deel maakt van de menus dus deze wordt niet in de vectoren gestopt.

- p1 is de tekst om aan te geven hoeveel keer speler 1 al heeft gewonnen.
- p2 is de tekst om aan te geven hoeveel keer speler 2 al heeft gewonnen.
- draws is de tekst om aan te geven hoeveel keer er al gelijk spel is geweest.
- Credits is de tekst om aan te geven door wie de applicatie werd gemaakt.
- Ennumeration is de tekst om aan te geven welke kolom door welk getal wordt voorgesteld.
- Message is de boodschap die geprint wordt bij het beëindigen van de executie van de applicatie.

3.3.4 Vectors adaptField

Hierin zitten de vectoren met offsets die gebruikt worden in een van de procedures adaptField afhankleijk van de vector.

- girdValuesVector hierin zitten de offsets van de arrays waar de elementen zitten die je in gridValues gaat veranderen afhankelijk van de grootte van het veld.
- gridDrawVector hierin zitten de offsets van de arrays waar de elementen zitten die je in grid gaat veranderen afhankelijk van de grootte van het veld.
- gridBorderVector hierin zitten de offsets van de arrays waar de elementen zitten die je in upperRightCorner, rowInBetween en firstTop gaat veranderen afhankelijk van de grootte van het veld.
- gridRowsVector hierin zitten de offsets van de arrays waar de elementen zitten die je in rowSeparation gaat veranderen afhankelijk van de grootte van het veld.
- gridCoorHorVector hierin zitten de offsets van de arrays waar de elementen zitten die je in horizontal gaat veranderen afhankelijk van de grootte van het veld.
- gridCooVertVector hierin zitten de offsets van de arrays waar de elementen zitten die je in vertical gaat veranderen afhankelijk van de grootte van het veld.
- gridEnnumVector hierin zitten de offsets van de arrays waar de elementen zitten die je in ennumeration gaat veranderen afhankelijk van de grootte van het veld.
- horWinVector hierin zitten de offsets van de arrays waar de elementen zitten die je in horCheck gaat veranderen afhankelijk van de grootte van het veld.
- vertWinVector hierin zitten de offsets van de arrays waar de elementen zitten die je in vertCheck gaat veranderen afhankelijk van de grootte van het veld.
- posWinVector hierin zitten de offsets van de arrays waar de elementen zitten die je in posCheck gaat veranderen afhankelijk van de grootte van het veld.
- negWinVector hierin zitten de offsets van de arrays waar de elementen zitten die je in negCheck gaat veranderen afhankelijk van de grootte van het veld.

3.3.5 Rules

Hierin zit de string die die de spelregels zal voorstellen. De enige constante hier is rules.

- rules is de grote uitleg van de spel regels die je kan terugvinden in de rules menu.

3.3.6 Constanten

Hierin zitten de constanten die gerbuikt worden doorheen de code. Onderandere vind je hier: grid, turnPiece, colors, vertical, horizontal, field, statusGrid, gridValues, winnerCount, pieceDim, rowInBetween,upperRightCorner en rowSeparation.

- grid zijn de constanten die nodig zijn om grafische weergave van het spelbord. Hun betekenis is als volgt: breedte van de contour, hoeveel plaats er tussen elk speelbaar plek is, hooghte van heet speelveld en de breedte van het speelveld.
- turnPiece zijn de die nodig zijn om een beurt op het scherm te kunnen aanduiden. Hun betekenis is als volgt: x-positie, y-positie en grootte van het muntstuk.
- colors zijn de constanten die nodig zijn om de kleuren op het scherm te kunnen representeren. Hun betekenis is als volgt: zwart, blauw, wit, geel en paars.
- vertical zijn de mogelijke posities in pixels wat betreft de hoogte.
- horizontal zijn de mogelijke posities in pixels wat betreft de breedte.
- field is de array die het speelveld zal representeren.
- statusGrid zal kunnen aangeven wat de staat van het spel is deze states zijn 2 aparte dingen die worden bijgehouden. In de eerste state ga je bijhouden wie er heeft gewonnen (speel verder, winnaar speler 1, winnaar speler 2 of gelijkspel). Bij de tweede ga je bijhouden of er gevraagd werd om de laaste zet ongdaan te maken(undo).

- gridValues zijn de dimenties van het speelveld. In dit geval is het speelveld 6 hoog en 7 breed.
- winnerCount zal gaan bijhouden hoeveel keer iedereen al heeft gewonnen in de huidige executie. De aantallen zijn als volgt: gelijkspel, speler 1 en speler 2.
- pieceDim is om aan te geven hoe groot een muntstuk op het speelveld zal zijn.
- upperRightCorner is om aan te geven waar het laatste element in field zich bevindt.
- rowInBetween is om aan te geven na hoeveel elementen in field je in een nieuwe kolom zit.
- rowSeparation zijn de waardes in field die aangeven waar de kolommen beginnen.
- Daarnaast zijn er ook niewe arrrays bijgekomen die elk hun doell is om elementen van de orginele constanten aan te pasen. Dit werdt al vermeld (zie sectie 3.3.4).

3.3.7 Interactions

Hierin zitten strings die geen titel zijn maar zijn de interacties die moglijk zijn om de menus te navigueren. Onderandere vind je hier: start, rules, back, exit, player1, player2, menu, pauze, restart, undo, resume, stats, alle mogelijke spelbordgroottes en move.

- start zal je terugvinden in mainMenu en kan je intrerageren door op “space” te drukken om het spel te starten.
- Rule zal je terugvinden in mainMenu en kan je interageren door op “r” te drukken om de regels op het scherm te laten zien.
- back zal je terugvinden in choiceMenu,rulesMenu en statisticsMenu en kan je interageren door op “b” te drukken om terugtekeren naar het voorige scherm.
- exit zal je terugvinden in mainMenu en announceMenu en kan je interageren door op “esc” te drukken om de applicatie te sluiten.
- player1 zal je terugvinden in choiceMenu en kan je interageren door op “1” te drukken om te bepalen dat speler 1 zal beginnen.
- player2 zal je terugvinden in choiceMenu en kan je interageren door op “2” te drukken om te bepalen dat speler 2 zal beginnen.
- Menu zal je terugvinden in announceMenu en kan je interageren door op “m” te drukken om terug naar mainMenu te gaan.
- pauze zal je terugvinden in gameMenu en kan je interageren door op “p” te drukken om het spel te pauzeren.
- restart zal je terugvinden in announceMenu en kan je interageren door op “e” te drukken om het spel te herstarten.
- undo zal je terugvinden in gameMenu en kan je interageren door op “d” te drukken om de laatste zet ongdaan te maken.
- resume zal je terugvinden in pauseMenu en kan je interageren door op “u” te drukken om het spel te hervatten.
- stats zal je terugvinden in mainMenu en announceMenu en kan je interageren door op “s” te drukken om naar de statistieken van de huidige exectutie te kijken.
- move zal je terugvinden in gameMenu en kan je interageren door op “1” tot en met “7” te drukken om een zet tee maken.
- Alle spelbord groottes worden getoond in een menu waar je de grootte van heet veld kan kiezen.

4 Problemen en oplossingen

Ik ben een paar moeilijkheden tegengekomen tijdens het programmeren. Voor sommigen heb ik een oplossing kunnen vinden voor anderen niet.

De eerste moeilijkheid waarmee ik te kampen had was het feit dat het assembly taal redelijk moeilijk te snappen en onduidelijk is als je geen ervaring ermee hebt. Enkele concrete voorbeelden: Door onervarenheid kreeg ik in het begin vaak errors. Eerst was het moeilijk te weten wat ze exact betekende en moest ik steeds op internet gaan zoeken naar hun betekenis. Naarmate ik gewend geraakt werden aan de taal, begon ik te weten wat welke error betekende en wist ik sneller waar ik moest gaan zoeken in de code naar een mogelijke fout.

Ik snapte eerst het verschil niet tussen mov en movzx. Ik wist niet wanneer welke gebruikt moest worden. Na een beetje zoeken ben ik te weten gekomen dat mov gebruikt wordt voor een 32 bit en movzx voor een 16 bit. En als je een 16 bit in een 32 bit wilt steken dan moet je de openstaande plaatsen vullen met nullen van daat de zx(zero extention).

Er zijn twee andere eigenschappen van assembly waaraan ik gewend moest raken in het begin. De eerste is dat de argumenten type sensitieve zijn en dus enkel en alleen een bepaalde type argument accepteren. Het tweede is bij het schrijven van procedures moet er "ret" gegeven worden op het einde anders crasht het programma, dit wist ik ook niet in het begin.

Ik ben naar de feedback sessie geweest en er werd mij meegegeven dat bij het aanmaken van de menu's ik veel te veel code dupliceer en dat ik dus een andere manier moest vinden om deze functionaliteit toch te implementeren zonder code duplicatie. Ook moest ik een betere manier vinden o de win conditie na te gaan want ik ging heel het veld door bij elke iteratie wat ervoor zorgt dat er wat performance blijft liggen.

5 Conclusie

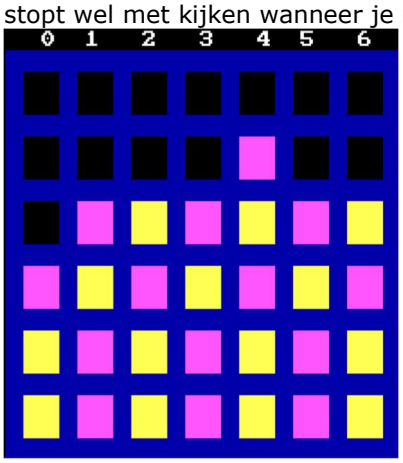
De feedback sessie was voor mij nuttig. Ik heb alle punten die werden vermeld verbeterd, zoals een groot deel van de code duplicatie, de win conditie en een extra functionaliteit toegevoegd.

Ik heb geprobeerd om zoveel mogelijk code duplicatie uit de code te halen maar op sommigen plaatsen blijft er toch wat herhaling, namelijk bij het gebruik van If statements door de code want het is nodig dat de executie van want er moet gebeuren nadat een conditie vervuld wordt steeds anders is. Soms is het ook zo dat je in plaats van een if een condtional (meerdere if genest) nodig hebt.

Ik heb ook had gewerkt om de codeduplicatie die je vond door de menus aan te maken om deze weg te krijgen. Dit is voor een groot deel gelukt. Nu bij het aanmaken van een menu roep je de procedure menuDisplay op die op zijn beurt 2 procedures omproept, namelijk: menuDistribution en menuConfiguration.

MenuDistribution zorgt ervoor dat je de juiste menu titel kan kiezen voor een bepaalde menu.
MenuConfiguration zorgt ervoor dat je de de rest van het menu goed kan laten verschijnen.
Het goede aan deze nieuwe versie is dat als je een nieuw menu wilt toevoegen dat je enkel de titel moet aanmaken en een paar simple aanpassingen moet maken aan menuConfiguration. Connect 4 wordt door menuDistribution geregeld en al de rest door menuConfuguration.

Een andere aanpassing die er gebeurd is, kan je zie bij het berekenen of er iemand gewonnen heeft (win conditie). Vroeger ging je bij elke zet het hele bord door om te zien of er iemand had h-gewonnen. Nu ga je bij elke nieuwe zet na de mogelijke vier op een rij af vanuit die positie. Je stopt wel met kijken wanneer je een munt tegen komt die niet meer van dezelfde speler is.



In dit geval is de laatste zet in kolom 4. Dus in dit geval zal de wincoditie eerst kijken of er iets horizontaal is, maar omdat hij kan zien dat er slecht 1 element is stopt die meteen. Vervolgens zoekt die verticaal waar die weer tot dezelfde conclusie komt. Dan zoekt je in de richting van f(x)=x en omdat de positie al zo hoog op het bord staat moet die naar beneden zoeken. Waar die snel tot de conclusie komt dat er vier op een rij staan en geeft het dan terug door de status in gameStatus aan te passen.
Stel dat die daar ook niks had gevonden dan zou die zoeken i de richting van f(x)=-x. Waar die tot de conclusie zou gekomen zijn dat er slecht 3 op een rij staan, maar omdat het niet voldoende is dan zou die meteen stoppen met zoeken want anders zoekt die een positie out of bounds.

Vervolgens hebt ik vermeld dat er een nieuwe functionaliteit aanwezig was en deze is dus het veranderen van spelbord grootte. Er is hiervoor ook een menu voorzien. Bij het drukken op “space” in de main menu kom je hier terecht waar je eerst een speelveld grootte kan kiezen. Al deze groottes zijn de officiele groottes van het spel(zie [connect four sizes]). Als je een andere exotische

grootte zou willen toevoegen dan kan dit gadaan worden door de volgende dingen aan te passen in de code. Een element in de volgende arrays te passen steeds op dezelfde positie: gridTickness, gridSpacing, gridHeight, gridWidth, gridVerticals, gridHorizontals, pieceDimentions, rowElements, corners, tops, startLastHor, startLastVert, startLastSlope, lastStartHor, lastStartVert, lastStartSlope, stepVertical, stepPos, stepNeg, validators, de tekst die in de menus word getoond, de ennumeratio die erbij hoort en stel dat je in de aanpassing meer als 8 rijen hebt en meer as 10 kollomen dan moet je ervoor zorgen dat er in vertical, horizontal en rowSeparation wel de juiste lengte hebben.

Een van de voordelen van de code is dat het redelijk leesbaar is, ook voor iemand die het programma niet heeft geschreven mits deze persoon al een basis ervaring heeft met assembly. Ook is de code zeer goed gedocumenteerd wel in het Engels maar nog steeds kan je waar nodig toch de nodige informatie uitlezen uit de commentaar.

Zoals vermeld in de sectie problemen, was het vooral in het begin dat ik moeite had omdat ik weinig tot geen ervaring had met assembly. Het is dus volkomen normaal dat de meeste problemen die ik had vooral in het begin voorkwamen. Dit project is echt een goede oefening om te leren coderen in assembly en ik heb enorm bijgeleerd. Vooral in het begin was het nodig om dingen te coderen in C en dan te zien hoe je dit zou omzetten in assembly.