VRIJE UNIVERSITEIT BRUSSEL

Project Computersystemen: Vier op een rij

Rodolfo Perez Tobar

December 26, 2022

Academiejaar 2022-2023

Bachelor Computerwetenschappen

Inhoudstafel

1 Inleiding 3
1.1 Intro 3
1.2 Spelregels 3
2 Handleiding 3
3 Programma 4
3.1 Sprite.asm 4
3.1.1 Datasegement 4
3.2 Setup.asm 4
3.2.1 Codesegment 4
3.2.2 Datasegement 5
3.3 Keys.asm 5
3.3.1 Codesegemnt 5
3.3.2 Datasegment 6
3.4 Mouse.asm 6
3.4.1 Codesegment 6
3.4.2 Datasegment 6
3.4.3 Global variables 6
3.5 Print.asm 7
3.5.1 Codesegement 7
3.5.2 Datasegement 7
3.6 Draw.asm 8
3.6.1 Global variables 8
3.6.2 Codesegement 9
3.6.3 Datasegement 9
3.7 Array.asm 10
3.7.1 Codesegement 10
3.7.2 Datasegement 10
3.8 Game.asm 11
3.8.1 Codesegement 12
3.9 Logic.asm 12
3.9.1 Codesegement 12
3.9.2 Datasegement 12
3.10 Menus.asm 12
3.10.1 Codesegement 12
3.10.2 Datasegement 12
3.11 Interact.asm 13
3.11.1 Codesegement 13
4 Problemen en oplossingen 14
5 Conclusie 14

© 2022-2023 Vrije Universiteit Brussel

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 terug op 0 gezet.
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 dat ligt in het interval [1,7] afhankelijk van op welk type bord je wilt spleen zal de bord grootte corresponderen met een van deze keuzes.
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. Ook is er de mogelijkheid om de andere speler te laten starten als men bij de keuze van de speler die begint zich heeft vergist door op ‘d’ te drukken wanneer het bord nog leeg is.
Let op, je kan enkel de getalen gebruiken die onder de functietoetsen zijn, de number pad is niet meer ondersteund in deze versie. 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 een van de nummer toetsen te drukken, de functietoetsen 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 “l” te drukken. Dit is zo gekozen om problemen bij de interpretatie van het ASCII-symbool voor ‘m’ anders is op een azerty dan op een qwerty toetsenbord. 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.
Een toevoeging tot dit programma is dat nu alle interacties die vroeger enkel konden gebeuren via het toetsenbord nu ook met de muis kunnen gebeuren door met de linkermuisknop te drukken op de gewenste knop op het scherm.

© 2022-2023 Vrije Universiteit Brussel

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. Ook is de code opgedeeld in verschillende files om de logica wat duidelijker te kunnen zien wat bij wat hoort. We zullen eerst kijken naar de globale omgeving vervolgens naar het codesegment en als laatste het datasegment. In elk van deze files. We zullen elk van de .ASM files overlopen.

3.1 Sprite.asm

In dit bestand zijn de Sprites gedefinieerd, het zijn gewoon variabelen die je kan aanspreken om deze sprites te tekenen. Dit bevat enkel elementen in het datasegment.

3.1.1 Datasegement

De sprites zijn: FieldXS, FieldS, FieldM, Field L, FieldXL, logo, statsIMG, ChoiseIMG en playerIMG.
Alle sprites met field* worden gebruikt om het speelveld te tekenen, deze zijn 1 square groot.
De andere sprites zijn logos die zichtbaar zijn doorheen de menu's.

3.2 Setup.asm

Dit bestand bestaat uit een codesegemnt en een datasegement, in dit bestand zitten de meeste elementen die een beetje overal in de code worden gebruikt.

3.2.1 Codesegment

Hierin kan je de procedures setVideoMode en terminateProcess 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.

© 2022-2023 Vrije Universiteit Brussel

3.2.2 Datasegement

Hierin zitten de constanten die gebruikt worden doorheen de code. Onder andere vind je hier: colors, vertical, horizontal, field, statusGrid, gridValues, firstTop, rowInBetween, upperRightCorner, currentMenu, fieldType, playerColor, movingSpace, moveDone, gridSpacing, validEntry, msg en rowSeparation.

- 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 state 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 laatste zet ongedaan te maken(undo).
- GridValues zijn de dimensies van het speelveld. In dit geval is het speelveld 6 hoog en 7 breed.
- 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.
- FirstTop houdt bij waar de linker bovenhoek van het veld zich bevindt.
- Msg is het bericht die je ziet verschijnen wanneer je de app verlaat.
- CurrentMenu houdt bij in welk menu je je bevind.
- FieldType houdt bij de keuze die je maakte voor de grootte van het veld.
- MovingSpace houdt bij waar je je huidig zet zal maken.
- MoveDone is een boolean die je helpt om te weten of je je zet hebt geplaatst.
- GridSpacing is een vector die elementen van grid helpen aan te passen.
- ValidEntry bakent het interval af voor de cijferinput in het spel zelf.
- PlayerColor houdt bij wie je had gekozen om te beginnen.

3.3 Keys.asm

Hier vind je alles wat te maken heeft met het gebruik van de keyboard als interactie. Hier ook werd het opgesplitst in datasegement en codesegemnt.

3.3.1 Codesegemnt

Hierin kan je de procedures _keyb_installKeyboardHandler, _keyb_unistallKeyboardHandler, keyboardHandler, delay numbersInput, numbersInputGame, keysMenuNavigation terug vinden.

- Install zorgt ervoor dat je de keyboardHandler juist op de stack zet.
- Unistall zorgt ervoor dat je deze er weer juist van de stack afhaalt.
- Delay zorgt ervoor dat het keyboard geen continue input leest en een beetje wacht voor dat de volgende input wordt gelezen.
- NumbersInput zorgt ervoor dat je de getallen in de menu's goed interpreteert.
- NumbersInputGame zorgt dat je de getallen van de kolommen juist interpreteert als en speler een zet wilt maken in het spel.
- KeysMenuNavigation regels alle interacties die mogelijk zijn in het spel door een combinatie te doen voor de getallen en letters.

© 2022-2023 Vrije Universiteit Brussel

3.3.2 Datasegment

Hierin vind je de variabelen die specifiek nodig zijn om je keyboard te kunnen werken. Onder andere: originalKeyboardHandlerS, originalKeyboardHandlerO, _keyb_keyboardState, _keyb_rawScanCode en _keyb_keysActive.

- OriginalKeyboardHandlerS houdt bij wat de vorige keyboardHandler was.
- OriginalKeyboardHandlerO houdt bij waar deze keyboardHandler in memory zat.
- _keyb_keyboardState houdt bij de state van elk van de Keys op je keyboard (ingedrukt of niet).
- _keyb_rawScanCode houdt bij wat de toets was die je ingedrukt had.
- _keyb_keysActive houdt bij hoeveel toetsen er tegelijk worden ingedrukt.

3.4 Mouse.asm

Hier vind je alles wat te maken heeft met het gebruik van je muis als interactie. Hier ook werd het opgesplitst in datasegement, global values en codesegement.

3.4.1 Codesegment

Hierin kan je de procedures mouse_present, mouse_internal_handler, mouse_install, mouse_uninstall, displayMouse, hideMouse, possibleNormalIntreaction, possibleNumberInteraction, possibleMoveInteraction, gameInteraction en buttonInteraction terug vinden.

- Mouse_present zorgt ervoor dat je kan nagaan of de muis effectief aanwezig is op het scherm.
- Internal_handler zorgt ervoor dat je een mouseRoutine kan gebruiken zonder de oude te verliezen.
- Install zorgt ervoor dat je mouseRoutine goed op de stack wordt gepusht.
- Unistall zorgt ervoor dat je mouseRoutine goed van de stack wordt afgehaald.
- PossibleNormalIntreaction zorgt ervoor dat je alle buttonInteraction dat geen getallen bevatten in je menu's goed kan afhandelen.
- PossibleNumberInteraction zorgt ervoor dat je alle buttonInteraction dat wel getallen bevatten in je menu's goed kan afhandelen.
- PossibleMoveInteraction zorgt ervoor dat je alle buttonInteraction dat een zet op het speelbord voorstellen goed kan afhandelen.
- GameInteraction zorgt ervoor dat je PossibleMoveInteraction met de juiste parameters correct kan oproepen.
- ButtonInteraction is de mouseHandler procedure die we in het spel gebruiken. Deze zal de correcte oproep doen van gameInteraction, possibleNumberInteraction en possibleNormalIntreaction.

3.4.2 Datasegment

Hierin zitten 2 variabelen. Namelijk: custom_mouse_handler en columnSpaces.

- Custom_mouse_handler houdt bij de offset van je procedure dat ervoor zorgt dat je muis op je spel werkt.
- ColumnSpaces houdt bij afhankelijk van de grootte van de sprites op welke hoogte in pixels je laatste rij begint.

3.4.3 Global variables

Hierin zitten 3 variabelen die nodig zijn voor de correcte interpretatie van alles wat op het scherm gebeurt.
Deze zijn: VMEMADR, SCRWIDTH en SCRHEIGHT.

- Vmemadr is het adres van waar je video geheugen begint.
- Scrwidth is de breedte van het scherm in pixels.
- Scrheight is de hoogte van het scherm in pixels.

© 2022-2023 Vrije Universiteit Brussel

3.5 Print.asm

Hier vind je alles wat te maken heeft met elementen printen op het scherm. Hier ook werd het opgesplitst in datasegement en codesegement.

3.5.1 Codesegement

Hier kan je de procedures moveCursor, printChar, printString en printScore terugvinden.

- MoveCursor zorgt ervoor dat je de cursor op het scherm kan verplaatsen naar de gewenste locatie.
- PrintChar laat je toe op een charter op het scherm te printen.
- PrintString laat je toe om een volledige string te printen op het scherm. Je kan ook newline printen door gebruik te maken van het symbool '~' en een string eindigt altijd met het symbool '$'.
- PrintScore laat je toe om de score op het scherm te tonen. Deze is wat special omdat het enkel om getallen gaat. Een toevoeging vergeleken met het oude versie si wel dat er nu geen limiet staat op hoeveel potjes je kan spelen, je kan nu maximaal zoveel representeren als je met 1 byte kun doen.

3.5.2 Datasegement

Deze is wat opgedeeld in variabelen, titels, interacties, game announcements, extra tekst en difficulties.
Alles dat niet tot de variabelen behoort zijn strings.

i. Constants

Onder Constants kan je de volgende variabelen in terug vinden: winnerCount, cursorPosVert en cursorPosHor.

- WinnerCount houdt bij wie hoeveel keer al heeft gewonnen (hoeveel keer gelijk spel, speler 1 en speler2).
- CursorPosVert is de hoogt van elk van de teksten die je moet afbeelden, deze worden voor gesteld in het aantal regels.
- CursorPosHor is de breedte waarop de eerst volgende karakter moet komen als je een getal hoger dan 9 moet afbeelden.

ii. Titels

Onder Titels vind je de titels die je doorheen de menu's ziet staan. Deze zijn: connect4, titlesRules, statistics, beginner, paused, difficulty en enumeration.

- Connect4 is de string "connenct 4" in de start menu.
- TitlesRules is de string "Rules" in de rules menu.
- Statistics is de string "Statistics" die je ziet in de statistics menu.
- Beginner is de string "chose who will begin:" die je ziet in de choise menu.
- Paused is de string "paused" die je ziet wanner je het spel pauzeert.
- Difficulty is de string "chose size of playfield:" die je ziet in de difficulty menu.
- Enumeration is de string met de getallen die voorstel waar welke kolom van je speel veld zich bevindt.

iii. Interacties

Onder interacties vind je alle strings die een knop op de menu's representeren. Onder andere vind je hier: start, rule, stats, exit, player1, player2, movment, pauze, undo, restart, menu, exitAfterPlay en statsAfterPlay.

- Start is de string die de knop startknop voorstelt.
- Rule is de string die de rules knop voorstelt.
- Exit is de string die de exit knop voorstelt.
- Player1 en player2 zijn de strings die de spelers knoppen voorstellen.
- Move is de string die je aangeeft welke knoppen je kan drukken om een zet te plaatsen in het speelveld.

© 2022-2023 Vrije Universiteit Brussel

- Pauze is de string voor de knop om het spel te pauzeren.
- Undo is de string die knop voorstelt om een zet ongedaan te maken.
- Restart is de string die de knop voorstelt om een nieuw potje te herstarten met dezelfde instellingen maar een nieuwe keuze over wie zal beginnen.
- Menu si de string die de knop voorstelt om terug te keren naar het hoofdmenu.
- ExitAfterPlay is de string die je nodig hebt om de knop exit voor te stellen, deze heeft een andere lengte dan Exit zelf.
- StatsAfterPlay is de string die je nodig hebt op de knop stats voor te stellen maar deze heeft een andere lengte dan stats zelf.

iv. Game announcements

Onder game announcements vind je de 3 elementen die je helpen om aan te geven wie er gewonnen heeft.
Deze zijn: winner, draw en turn.

- Winner is de string die gebruikt wordt om aan te geven wie de winnaar is.
- Draw is de string die gebruikt wordt om aan te geven wie dat er gelijk spel is.
- Turn is de string die gebruikt wordt om aan te geven wie nu aan de beurt is.

v. Extra tekst

Onder extra tekst vinden we tekst die niet onder categorie te classificeren valt. Hier vind je: p1, p2, draws en credits.

- P1 is de string die zal gebruikt worden om aan te geven hoeveel keer speler 1 heeft gewonnen.
- P2 is de string die gebruikt wordt om aan te geven hoeveel keer speler 2 heeft gewonnen.
- Draws is de string die gebruikt wordt om aan te geven hoeveel keer er gelijk spel is geweest.
- Credits is de string die aangeeft wie de eigenaar is van dit project.

vi. Difficulties

Onder difficulties vinden we de strings die nodig zijn om de verschillenden levels aan te geven als keuze in de menu's. Hier vind je: veasy, easy, standard, tricky, hard, extreme en square.

- Veasy is de string die aangeeft dat het speelveld dat je gaat kiezen van grootte 5*4 is.
- Easy is de string die aangeeft dat het speelveld dat je gaat kiezen van grootte 6*5 is.
- Standard is de string die aangeeft dat het speelveld dat je gaat kiezen van grootte 7*6 is.
- Tricky is de string die aangeeft dat het speelveld dat je gaat kiezen van grootte 8*7 is.
- Hard is de string die aangeeft dat het speelveld dat je gaat kiezen van grootte 9*7 is.
- Extreme is de string die aangeeft dat het speelveld dat je gaat kiezen van grootte 10*7 is.
- Square is de string die aangeeft dat het speelveld dat je gaat kiezen van grootte 8*8 is.

3.6 Draw.asm

Hier zal je alle elementen terugvinden die te maken hebben met objecten tekenen op het scherm. Op nieuw zijn hier 3 delen op te speuren. Global variabels, codesegment en datasegment.

3.6.1 Global variables

Hierin zitten 3 variabelen die nodig zijn voor de correcte interpretatie van alles wat op het scherm gebeurt.
Deze zijn: VMEMADR, SCRWIDTH en SCRHEIGHT.

- Vmemadr is het adres van waar je video geheugen begint.
- Scrwidth is de breedte van het scherm in pixels.
- Scrheight is de hoogte van het scherm in pixels.

© 2022-2023 Vrije Universiteit Brussel

3.6.2 Codesegement

Hierin kan je de procedures fillBackground, drawRectangle, drawGrid, drawMove, makeButton, playerTurn, announceInfo, drawSprite, drawer, drawLogoDistribution 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 sinds een rechthoek een gearceerde rechthoek 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 rood 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.
- DrawSprite zorgt ervoor dat je een sprite correct op het scherm laat verschijnen. De input zijn: x-coördinaat, y-coördinaat, de gewenste sprite, breedte van de sprite, hoogte van de sprite, hoe moet je de zwarte pixels invullen en of je een mask sprite wilt tekenen of niet.
- Drawer zorgt ervoor dat je de field sprites correct kan tekenen.
- DrawLogoDistribution zorgt ervoor dat je de logos goed kan tekenen.
- AnnounceInfo zorgt ervoor dat je na afloop van een spel de juiste winnaar kan aangeven als er iemand gewonnen heeft. Als er gelijk spel was dan geeft je dit ook aan.

3.6.3 Datasegement

Hierin zitten de constanten die gebruikt worden doorheen de code. Ook kan je zien dat er hier een opsplitsing is in 2 categorieën. De constanten en vectoren die je nodig hebt om iteraties te kunnen uitvoeren.

i. Constants

Hier vind je: grid, turnPiece, pieceDim, buttonSize, logoDimentions, LogoStartPoint en logoYValue.

- Grid zijn de constanten die nodig zijn om grafische weergave van het spelbord. Hun betekenis is als volgt: hoeveel plaats er tussen elk speelbaar plek is.
- 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.
- PieceDim is om aan te geven hoe groot een muntstuk op het speelveld zal zijn.
- ButtonSize houdt bij dimensies om een knop op het scherm te laten verschijnen. Hun betekenis is als volgt: lettergrootte, letterbreedte, hoogte van de knop, breedte van een normale knop, breedte van een kleinere knop.
- LogoDimentions houdt bij wat de dimensie is van elke logo.
- LogoStartPoint houdt bij de x-waarde van waar elke logo zich bevindt.
- LogoYValue houdt bij wat de y-waarde is van elke logo.

ii. Vectors

Hier vind je de vectoren die nodig zijn om bepaalde iteratieve procedure te kunnen uitvoeren.
Deze zijn: Sprites, logos en logoPlace.

- Sprites houdt en vector bij met de sprites die nodig zijn om het spel te spelen.
- Logos houdt alle logos bij in een array om over deze te kunnen itereren.
- LogoPlace houdt bij de informatie die je nodig hebt om je logos correct te kunnen plaatsen.

© 2022-2023 Vrije Universiteit Brussel

3.7 Array.asm

In dit bestand kan je alles terug vinden die te maken heeft met array manipulaties. Dit bestand bestaat uit 2 delen. Codesegement en datasegement.

3.7.1 Codesegement

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 correcte 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 wanneer je het spel pauzeert en dan 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.
- AdaptCoordinates 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 enumeration.
- 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 bevindt zich in validEntry.
- 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 procedure kan aanroepen om zo de code gemakkelijker te kunnen onderhouden.

3.7.2 Datasegement

Hier vind je de variabelen die nodig zijn om de array manipulaties te kunnen voltooien. Hier zijn de elementen ook opgedeeld in constants en vectoren.

i. Constanten

Hier zijn de constanten terug te vinden. Onder andere *check, game*, ennum*, grid*, r*, tops, rowElements, corners, validators, v*, h*, pieceDimentions, startLast*, lastStart*, position* en steps*.

- *Check zijn de warden die je nodig hebt om een check in 1 richting te kunnen uitvoeren. Respectievelijk is de richting aangeven in de naam.
- Game* zijn de strings respectievelijk die je nodig hebt om aan te geven in het spel welke toetsen je mag indrukken om een zet te plaatsen. Deze hangen af van de keuzen die je hebt gemaakt in het difficulty menu.

© 2022-2023 Vrije Universiteit Brussel

- Ennum* zijn de strings die je nodig hebt om de juiste kolommen te kunnen aanduiden in het spel. Deze hangen af van de keuze die je gemaakt hebt in difficulty menu.
- Grid* zullen de elementen in gridValues gaan aanpassen om een correct bord te kunnen bekomen.
- R* zijn de elementen die je nodig hebt om rowSeparation goed te kunnen aanpassen.
- Tops zorgen ervoor dat je firstTop goed kan aanpassen.
- RowElements zorgt ervoor dat je rowInBetween goed kan aanpassen.
- Corners wordt gebruikt om upperRightCorner goed aan te passen.
- Validators wordt gebruikt om validEntry goed aan te passen.
- V* worden gebruikt om vertical aan te passen.
- H* worden gebruikt om horizontal goed aan te passen.
- PieceDimentions worden gebruikt om pieceDim goed aan te passen.
- StartLast* worden gebruikt om in elke check* het eerste element te vervangen. Elk respectievelijk met hun afkorting.
- LastStart* worden gebruikt om het tweede element te vervangen in elke check respectievelijk. Slope wordt hier wel gebruikt om de twee diagonalen aan te passen.
- Position* wordt gebruikt om het derde element aan te passen in elke check. Horizontaal, verticaal en stijgende rechte maakt gebruik van postionHor, en de dalende rechte gebruikt het ander.
- Steps* worden gebruik om de vierde paramater van de check aan te passen.

ii. Vectors

Hier kan je de vectoren terugvinden die je zal gebruiken om de iteraties uit te voeren en de aanpassing te kunnen maken. Je vindt hier: gridValuesVector, gridDrawVector, gridBorderVector, gridRowsVector, gridCooHorVector, gridCooVertVector, horWinVector, vertWinVector, poswinVector, negWinVector, gridEnnumVector en moveDisplayVector.

- GridValuesVector 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.
- GridCooHorVector 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 enumeration 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.
- MoveDisplayVector hierin zitten de offsets van de arrays waar de elementen zitten die je movement gaat aanpassen om de tekst correct te tonen afhankelijk van het speelveld.

3.8 Game.asm

Dit is het startpunt om je programma op te starten, in C wordt dit gezien als main. Ditt bestand bestaat enkel uit een codesegement.

© 2022-2023 Vrije Universiteit Brussel

3.8.1 Codesegement

Hier zit enkel de procedure main.

- Main is het startpunt van het programma, deze heb je nodig om het programma te runnen.

3.9 Logic.asm

In dit bestand zit alles wat te maken heeft met het spel logica. Dit bestand bestaat uit 2 delen. Het codesegement en het datasegement.

3.9.1 Codesegement

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

- MakeMove zorgt ervoor dat je een zet op het bord kan spelen.
- FullCheck zal gaan kijken of het bord volledig gevuld 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.
- CheckWinner maakt gebruik van checkWinForDirection om zo het heel veld af te gaan en op zoek te gaan naar één mogelijke 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 gerepresenteerd: 0 betekent speel verder,1 betekent speler 1 heeft gewonnen, 2 betekent speler 2 heeft gewonnen en 3 betekent er is gelijk spel.

3.9.2 Datasegement

In dit onderdeel zit een vector die je nodig hebt om na te gaan of er iemand heeft gewonnen. Deze is winChecker.

- Wincheckere bevat de offsets van de 4 vectoren die je nodig hebt om na te gaan of er een winnaar was in 1 van de 4 richtingen.

3.10 Menus.asm

In dit bestand zit alles wat te maken heeft met de menu's. Dit bestand is opgedeeld in 2 grote delen. Codesegement en datasegement. In het datasegement vind je wel 3 grote onderdelen: vectoren, rules en special interacties.

3.10.1 Codesegement

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(titel) 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.

3.10.2 Datasegement

Hier vind je alle elementen die nodig zijn om de menu's correct af te beelden

© 2022-2023 Vrije Universiteit Brussel

i. Vectors

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

- TextHeader is de vector met alle titels 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 announcements 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.

ii. Rules

Hier zit enkel de variabele die de spelregels zal tonen op het scherm.

- Rules is de string die de spelregels bevat.

iii. Special interations

Hier zitten een paar strings die je constants doorheen alle menu's weer ziet terug duiken. Hier door zijn deze wat speciaal van aard. Deze zijn: resume, back en moving.

- Resume is de string die de resume knoop moet voorstellen.
- Back is de string de die de back button moet voorstellen.
- Moving is de string die aangeeft dat een zet ook met de muis gedaan kan worden.

3.11 Interact.asm

In dit bestand vind je alle terug die te maken heeft met de interacties van het spel. Hier zit de eigenlijke gameloop van het spel. Deze bestaat enkel uit het onderdeel codesegement.

3.11.1 Codesegement

In dit onderdeel vindt zicht het eigenlijke spel. Alles komt hier samen om een goed werkend spel te vormen. Hier 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.

© 2022-2023 Vrije Universiteit Brussel

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 daar de zx (zero extension).

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 om 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.

Na een succesvolle upgrade van het project heb ik zelf wat meer feedback gevraagd om te zien waar ik aspecten kon verbeteren. Ik kreeg toe te horen dat ik 5 dingen zou kunnen doen. Zorg ervoor dat de muis ook werkt in het spel, maak gebruik van sprites, splits alles op in meerder files, implementeer animaties in het spel en maak zelf een AI tegenstander. Na lange overweging heb een paar van deze elementen uitgekozen. Ik heb besloten om meerder files te maken zoals het nu wat duidelijk is vanuit deel 3. Ik heb ook besloten om de muis te laten gebruiken in het spel zelf en als laatste heb ik ook sprites gebruikt in mijn spel.

Om de meerder files te kunnen gebruiken was het nodig om de makefile wat aan te passen. De traditionele makefile laat je enkel toe om maximaal 5 files te gebruiken, maar met de aanpassing kan je er zoveel maken als dat je wilt.

Het gebruik van de muis was ook lastig omdat je weinig informatie om het internet kan vinden over de bepaalde interrupts die je nodig hebt om elementen te kunnen doen met je muis. Gelukkig zit er in Exempels een voor beeld van hoe de muis werkt, na grondige studie kwam ik tot de conclusie dat ik mijn huidige code moest aanpassen om zowel keyboard en muis te ondersteunen. Hierdoor is zijn er dus 2 files bijgekomen (mouse.asm en keys.asm).

Om de sprites op het veld te tekenen heb ik gebruik gemaakt van de tekst modus, ik maak dus zelf mijn sprites aan in string formaat en lees dezen dan in om zo mijn sprites te bekomen. Dit is niet ideaal, maar het werkt wel. De beste oplossing zou zijn om .bin files te gebruiken om op deze manier minder geheugen te gebruiken en zo ook een wat meer modulair project te hebben.

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 extra functionaliteit toegevoegd. Jammer genoeg ben ik niet zover geraakt om animaties te kunnen maken nog een Ai tegenstander te implenteren.

© 2022-2023 Vrije Universiteit Brussel

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-statements genest) nodig hebt.

De toevoeging van muis en sprites maken van dit project een zeer redelijk afgewerkt project. Er zijn natuurlijk wel aspecten die ontbreken zoals een Ai tegenstander, animatie en muziek om dit als compleet te kunnen beschouwen.
Maar dit haalt niet weg dat je een zeer afgewerkt resultaat behaalt met wat er nu al is.

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.

© 2022-2023 Vrije Universiteit Brussel