CHESS.EXE

Schaken in 32-bit assembly.

Lender Lismond Vincent Mostert

2019-12-22

Inhoudsopgave

Introductie

Dit project simuleert een virttueel schaakbord, met een validatie van de basisregels.

Gebruik

Het project kan gewoon gecompileerd worden met `wmake`. Dit produceert een executable: `CHESS.EXE`. Als deze executable gestart is wordt er meteen een nieuw spel gestart. De eerste beurt is altijd aan zwart, die het selectievakje kan rondbewegen. De toetsen om te bewegen zijn: W, A, S, D of H, J, K, L op een qwerty-toetsenbord. Als het selectievakje over het gewenste stuk staat, duw dan op spatie of enter om dit stuk te selecteren om te bewegen. Over het gekozen stuk blijft een selectievak staan, en men kan nu een tweede vakje rondbewegen om de bestemming van het stuk te selecteren. Duw opnieuw op spatie of enter om de selectie te bevestigen. Als een geldige verplaatsing is geselecteerd dan zal

het stuk verplaatsen en is het de beurt aan wit. Duw op escape om het spel af te sluiten.

Datatypes

Piece

Dit stelt een schaakstuk op het bord voor, en is een byte groot:

Naam	Grootte	Waarde
Color	1	0 = zwart, 1 = wit
Kind	7	0 = leeg, 1 = koning, 2 = koningin, 3 = toren, 4 = paard, 5 = loper, 6 = pion

Een leeg schaakstuk kan eenderwelke kleur hebben.

Position

Dit stelt een positie op het schaakbord voor en is een woord groot:

Naam	Grootte	Waarde
X	8	x-coördinaat van de positie op het bord, beginnende bij 0.
Y	8	y-coördinaat van de positie op het bord, beginnende bij 0.

Zowel x en y hebben dus als waarde 0 t.e.m. 7.

Ontwerp

De code van dit project is verdeeld over drie bestanden:

- Het *DRAW.ASM* bestand staat in voor alles wat getekend of aangeduid moet worden in het schaakspel. Hier worden dus de vakjes getekend, de figuren en de te aanduiden vakjes worden hier aangeduid.
- In het bestand *ENGINE.ASM* staat de motor van het spel. Dit bestand omvat de functionaliteit en regels die nodig zijn het spel te kunnen spelen. Alle logica zit hier dus in verwerkt.
- De echte gameloop zit in `CHESS.ASM`. Hier worden alle andere procedures opgeroepen om het spel te maken en worden de toetsenbord-instructies ingelezen. Afhankelijk van deze input wordt er steeds andere output gegenereerd.

De interfce tussen de engine en het grafische gedeelte is een array van bytes waarin de stukken zijn opgeslagen. In de game-loop wordt na iedere zet het spel opnieuw getekend door deze array uit te lezen.

CHESS.ASM

Configuratie

Allereerst wordt de video mode op *13h* gezet, vervolgens word het kleuren- pallet ingeladen. Daarnaast wordt ook de achtergrond ingesteld op VUB-blauw. Na de achtergrond te initialiseren kunnen we het bord tekenen en de start- positie aanduiden.

Gameloop

Nu we alles geïnitialiseerd hebben, kunnen we beginnen met input te lezen. Ten eerste wordt er gekeken of er een *keystroke* plaatsvindt. Wanneer dit niet is gebeurd, wordt de gameloop terug gestart en wordt er weer input ingelezen. Stel, er is een toets ingedrukt, dan zal deze check wel kloppen en zal de inhoud van deze toets opgevraagd worden. Vervolgens wordt deze vergeleken met de toetsen die ingedrukt kunnen worden. Als deze met geen toets overeenkomt betekent dit dat een verkeerde toets is ingedrukt en zal wederom, de gameloop herstarten. Anderzijds zal deze de procedure oproepen die geassocieerd is met die toets.

DRAW.ASM

Configuratie

Om het bord te tekenen is het noodzakelijk een paar zaken eerst te configureren zoals de *video_mode*, de *background* en het *palette*.

Bord tekenen

Wanneer we alles hebben geconfigureerd kunnen we beginnen met het effectief maken en tekenen van het bord. Dit houdt in dat we vierkanten moeten kunnen tekenen, deze moeten kunnen gebruiken om het bord te tekenen en tot slot de stukken kunnen tekenen bovenop deze vierkantjes.

Draw_square

```
@@color:dword, @@x:dword, @@y:dword
```

De procedure zal de meegegeven x- en y-coördinaat gebruiken als linkerbovenhoek van het vierkant. Eerst wordt het beginpunt van het vierkant berekend en vervolgens worden er recursief lijnen getekend met een meegegeven waarde in *SQUARE_SIZE*. Deze constante gebruiken we dus ook voor de diepte.

Draw_piece

```
@@piece:dword, @@x:dword, @@y:dword
```

Tekent een schaakstuk op het bord. De bitmaps van de assets worden gegenereerd met `mkassets.sh` op een UNIX-systeem met python 3. Die worden dan in de binary geïntegreerd, de assets zijn klein genoeg om dit mogelijk te maken.

`draw_piece` leest de bitmap uit een label en kopieert de bytes die niet nul zijn naar de framebuffer. De lege bytes in de bitmap bepalen dus het alfakanaal.

Draw_board

Deze procedure maakt twee lokale variabele aan: *@@x:dword, @@y:dword*. Deze worden bij elke iteratie veranderd en meegegeven aan *draw_square*. Er wordt ook een extra counter bijgehouden in de loop zodat we kunnen weten of we een wit of een grijs vakje moeten tekenen.

Selecteren

Naast het bord tekenen moet het schaakspel ook de mogelijkheid hebben vakjes te selecteren, dit omdat de speler wilt weten welke pion hij/zij vast heeft. Ook is dit handig voor de mogelijke zetten aan te duiden.

Set_selection

Voor deze procedure introduceren we een paar globale variabelen die constant aangepast zullen worden. Zo hebben we *CURRENT_MODE*, die de huidige mode toont, dit kan of `mode_from` zijn of `mode_to`. Vervolgens hebben we de *FROM_X, FROM_Y en TO_X en TO_Y* variabelen. Hier zullen de coördinaten ingestoken worden van de gewenste *from en to* vakjes. `set_selection` zal dus kijken of de variabelen al aangepast zijn en als dit niet is gebeurd dan zullen deze aangepast worden. De registers eax en ebx houden de *from en to* posities bij in *STRUC position* formaat.

Move_selection

```
@@dx:dword, @@dy:dword
```

Beweegt het huidige selectievakje rond. Deze procedure bepaalt dus zelf of dat het oorsprong- of bestemmingsvak moet zijn. Op de oude plaats van het vakje wordt ook een nieuwe rand getekend.

Highlight_square

```
@@x0:dword, @@y0:dword, @@color:dword
```

De procedure *highlight_square* lijkt sterk op de procedure *draw_square*. Dit omdat we in theorie een vierkant tekenen, het verschil hier is dat we enkel de rand tekenen in tegenstelling tot *draw_square*, waar we een vol, vierkant tekenden. Merk op dat we bij *draw_square* als x- en y-coördinaten pixels meegaven, hier geven we indices mee. (*highlight_square, 7, 7, 3h*). Het verschil in procedures zit hem in het naar beneden tekenen, hier tekenen geen lijn maar twee punten. Het eerste punt staat gelijk aan de x-coördinaat en het tweede punt staat gelijk aan de x-coördinaat opgeteld met de grote van een vierkant min een.

Highlight_all

Dit is de procedure die alle te aanduiden vakjes gaat aanduiden door steeds *highlight_square* op te roepen met de juiste waarden.

Reset_highlight

Deze procedure zet de globale variabelen in het datasegment terug naar de standaard waarden en de mode naar *mode_from* zodat we weten welke kleur we moeten gebruiken.

ENGINE.ASM

Bewegen

Do_move

```
@@move_from:dword, @@move_to:dword
```

Beweegt een vakje in de interne representatie van de engine. In `eax` wordt bewaard of de verplaatsing geldig was. Naast het verplaatsen zelf wordt ook gevalideerd dat de zet geldig is. Om dit te doen worden eerst enkele algemene validaties gedaan, zoals het controleren dat een schaakstuk de juiste kleur heeft, en dat de speler zijn eigen stuk niet pakt. Daarna wordt ook nog eens per schaakstuk gekeken of de beweging geldig is, en indien relevant, of er schaakstukken in de weg staan. Om de juiste code uit te voeren voor het juiste schaakstuk is er een jump table van procedures die in `eax` nul zetten als de zet ongeldig is.

Is_not_empty

```
@@tile:dword
```

Deze hulpprocedure zal kijken of het vakje met de meegegeven tegel bezet is of niet. Wanneer de er op die meegegeven tegen een pion staat, zal deze procedure een 1 in het eax register steken en teruggeven. Anders wordt er een nul in gestoken.

Validate_chess_piece

```
@@move_from:dword, @@move_to:dword, @@piece_from:dword,
@@piece_to:dword
```

Deze procedures zijn gelijkaardig aan elkaar, de enige verschillen zitten in de logica. Elk schaakstuk heeft andere rechten en restricties. Daarom worden eerst alle regels afgegaan om te zien of het schaakstuk wel deze zet mag zetten. Vervolgens wordt er gekeken of er geen obstakels in de weg naar die zet zijn. Wanneer deze op het einde van de *validate* geraakt, betekent dit dat het een geldige zet is en wordt het eax register overschreven met waarde 1, anders met 0.