

Komplikationer med skærmopløsning i et 2D/3D miljø

David Recke, Emil Vad, Simon Vinther

March 27, 2015

Contents

1	Introduktion	1
1.1	Abstract	1
1.2	Forord	1
2	Aspekt ratio	4
2.1	Field of View	4
2.2	Letterbox	5
2.3	Virtual viewport	5

1. Introduktion

1.1 Abstract

derbderb gorg gorg

1.2 Forord

"testing subfiles" whup whup

Teksture skallering Når laver sit spil til forskellige skærmopløsninger skal man kunne skaller sine teksture og sprites man kan enten skaller op eller ned. Så det første man tænker kunne være en god ide, er at lave sine teksturen i rigtig høj opløsning så du er sikker på at de har en godt kvalitet når de er på en højre opløsning, også bare skaller det ned på lavere opløsninger. Der bare et problem når man skaller, da du ikke altid kan skaller ned ordenligt, hvis du f.eks. skaller fire pixels ned til det halve for du to men når du skaller fire pixels ned til en tredjedel for du 1,333 og når du ikke kan havde halve pixels er det inden 2 eller 1 hvilket kan få grafikken til at se aliased ud. Mange morderne eniges og billede programmer prøver at slev at løse problemet ved søger for at bløvere det sammen så det er mindre synligt med det kan stadig være et problem ud over det tager det også lang tid at lave en teksture / sprites med meget høj opløsning. En anden løsning er at lave dine med den lavest opløsning for dine teksture / sprites kan havde derefter skaller dem op så det for normalt bare dine spirtes til at se pixlerne ud. En tredje måde at gøre det på er at lave vektor grafik da de skaller bedst med det tager længer tid at lave vektor grafik end normal grafik, ud over det tager vektor grafikken mere cpu end normal grafik men mindre ram derfor er der nogen pladformer som vector grafik er bedrer da cpu kan varerier end dele fra platform til platform. Den sidste løsning som mest kan bruges i 3d spil er at lave tileable teksturer da det tager korter tid at lave også bruger det værken meget ram eller cpu de kan stadig løbe ud i problem med at det ser aliaseret ud når det nedskalles med da ikke tager lang tid om juster på en tilable teksturen kan man indstille den meget indtil den ser godt ud. Ikke alle teksturer kan være tilable da det nogen gange ødelægger illusion da man kan se at teksturen gentager sig selv. Det er ikke en bedst måde at løse teksture og sprites skallering problemet på da det afhænger meget af spillet og tid man har til at lave det. Generelt er det en god idé at lave høj opløsning da mange eniges søger for at det stadig ser ok ud i lavere resolution og tilealbe teksturer være gang man kan.

Listing 1.1: Et eksempel på en kamera klasse

```

1 //a struct to repsent a point in a 2 aksese
2 struct Point
3 {
4     public float x = 0;
5     public float y = 0;
6 }
7
8 class Kamera
9 {
10    //The Position of camera in game world koordinate
11    Point postion;
12
13    int screenHeight;
14    int screenWidth;
15
16    // the amount of x and y game world koordinate the kamera can see
17    static const float maxXOnScreen = 100;
18    static const float maxYOnScreen = 100;
19
20    Kamera(Point postion, int screenHeight, int screenWidth)
21    {
22        this.postion = postion;

```

```

23     this.screenHeight = screenHeight;
24     this.screenWidth = screenWidth;
25   }
26
27   Point PostionOnScreen(Point postion)
28   {
29     Point newPostion;
30     newPostion.x = screenWidth/2 +
31       ( (postion.x - this.postion.x)* screenWidth/maxXOnScreen );
32     newPostion.y = screenHeight / 2 +
33       ((postion.y - this.postion.y) * screenWidth / maxXOnScreen);
34     return newPostion;
35   }
36
37 }
```

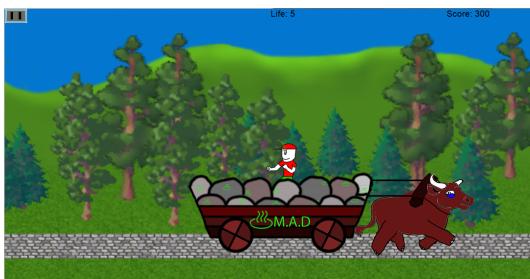


Figure 1.1: MAD i produktions opløsning



Figure 1.2: MAD i 1280x800 opløsning med skærm kordinator



Figure 1.3: MADscaling3



Figure 1.4: MADscaling4

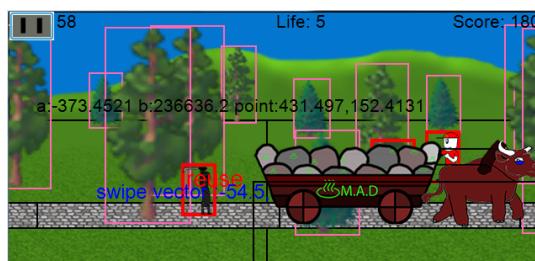


Figure 1.5: derb

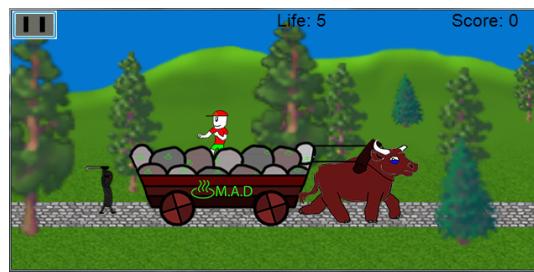


Figure 1.6: derbo

2. Aspekt ratio

Aspekt ratio refererer til den proportionelle forskel mellem en skærms højde og brede [4]. Der findes mange forskellige aspekt ratios, som bliver brugt til forskellige formål [7]. I dag er de langt mest benyttede opløsning blandt spillere på Steam 1920x1080, efterfulgt af 1366x768. Begge disse opløsninger er 16:9 formater. Andre formater der udgør en betydelig del er 1280x1024 (5:4), 1680x1050 (16:10), og 1440x900 (16:10) [6].

At skifte mellem to forskellige opløsninger, som deler det samme aspekt ratio, er relativt simpelt, da teksturer blot kan skaleres til at have en anden størrelse. Så længe at ratioen er den samme, og teksturen bliver skaleret ned til en dårligere opløsning, er der generelt ikke et problem.

Problematikken opstår når at aspekt ratioen ændres, da en skalering ikke længere blot vil ændre på størrelsen af teksturen, men også på forholdet mellem bredden og højde. Det betyder at akserne ikke bliver skaleret synkront, hvilket kan give billedet et andet visuelt indtryk. Dette har været tilfældet i mange ældre spil, der originalt ikke var lavet til widescreen, blandt andet Warcraft 3, som kan ses på figur 2.1 og 2.2. [8]



Figure 2.1: Warcraft 3 i dets originale 4:3 aspekt ratio



Figure 2.2: Warcraft 3 skaleret asynkront til et 16:9 aspekt ratio

2.1 Field of View

Der er to måder at komme uden om dette problem på. Den ene er at ændre på field of view (FOV), som er hvor stor en del af spilleren ser af spilverdenen. Dette bringer imidlertid sine egne problemer på banen. Når FOV ændres, ændres der også på det output spilleren får på sin skærm, hvilket betyder at alle spillere ikke ser spilverdenen ens. Hvis at spillet ikke er kompetitivt spil, behøver dette ikke at være en bekymring, da spilleren kun spiller mod sig selv, og spillet i den forstand er ligestillet for alle. Hvis spillet derimod er kompetitivt, medfører det med stor sandsynlighed at spillere med et vist aspekt ratio har en fordel over dem med et andet aspekt ratio [2].

Hvis vi ser på et 2d spil, og forestiller os et endless runner spil, så som Robot Unicorn Attack, hvor at man kun ser hvad der er lige foran sin karakter, vil et bredere FOV betyde at man har længere tid at forberede sig på de forhindringer der kommer forude.

Hvis vi ser på et 3d spil, så som CS:GO, vil et bredere field of view betyde at spilleren kan se længere til siderne. Det vil sige at spillere med et mindre FOV effektivt set har en blind vinkel, som at deres modspillere med et bredere FOV ikke har, hvilket giver en stor fordel når det kommer til at få visuel kontakt med sine fjender. På figur 2.4 ses f.eks. en dør til højre, som ikke kan ses på figur 2.3 pga. forskellen i FOV.



Figure 2.3: CS:GO med aspekt ratio 4:3



Figure 2.4: CS:GO med aspekt ratio 16:9

2.2 Letterbox

Den anden måde at komme uden om skaleringsproblemer på, er at tilføje sorte bars, også kaldet letterbox, til toppen og bunden af skærmen. På den måde tvinges spillet reelt set til at køre i et statisk aspekt ratio, og dermed er der ingen uretfærdigheder når det kommer til FOV [5]. Disse sorte bars kan også blive smidt på højre og venstre side af skærmen, og i disse tilfælde kaldes de pillarboxing [1]. Nogle spil er nødt til at benytte sig af dette koncept, for at deres gameplay kan hænge sammen.

Et af disse spil er Speedrunners. Formålet med Speedrunners er at løbe om kap, og komme fremad hurtigst muligt, og dermed efterlade ens modstander bag sig. Alle fire spillere ser på den samme del af spilverdenen, og når en spiller falder for langt bagefter, og kommer uden for skærmen, bliver spilleren elimineret. I det at spillet afhænger af at lose condition er at komme uden for skærmen, ville forskellige aspekt ratioer betyde at dem med et mindre FOV enten ville dø tidligere, eller vil være ude af stand til at se sin karakter mens han stadig er i live. Ved at tilføje letterboxes til spillere der ikke kører med et 16:9 aspekt ratio, sikrer spillet sig at alle har det samme syn på spil verdenen, og der er enighed om hvornår en spiller er ude af skærmen. På figur 2.6 ses spillet i det tiltænkte 16:9 aspekt ratio, og på figur 2.5 ses spillet i et 4:3 ratio hvor det er letterboxed.

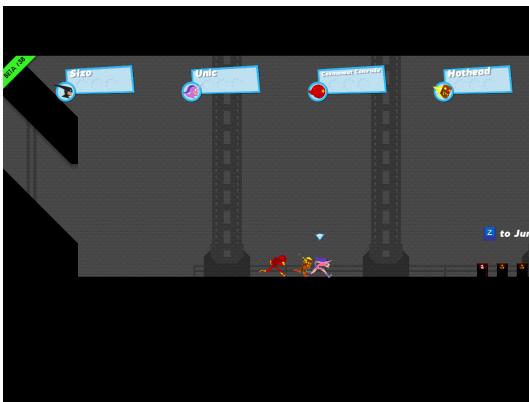


Figure 2.5: Speedrunners med aspekt ratio 4:3

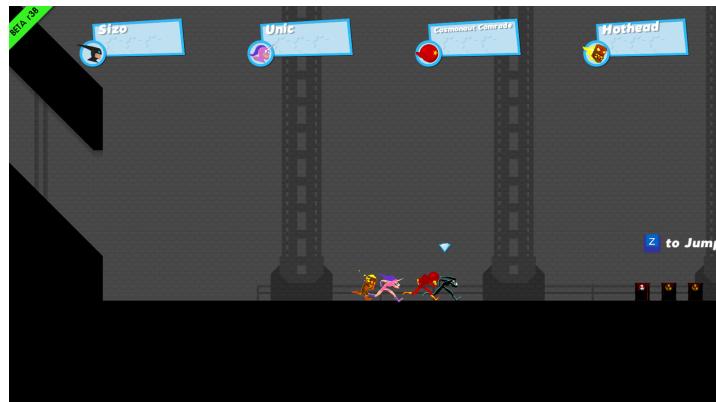


Figure 2.6: Speedrunners med aspekt ratio 16:9

2.3 Virtual viewport

På mobilmarkedet findes der ligeledes mange forskellige aspekt ratios, og det bearbejdes i store træk på samme måde som på platforme med større displays. Det er dog i de færreste tilfælde at det er ønskeligt at tilføje letterboxes til mobil platformen, da der i forvejen ikke er meget plads at arbejde med, og det i desuden kan give et indtryk af at spillet ikke er beregnet til den brugte model.

En virtual viewport kan bruges til at vise en del af et environment, der kan justere sig selv efter dit aspekt ratio. Ved at lave en baggrund bred nok til at supportere et bredt aspekt ratio, og højt nok til at supportere et smalt et, kan man ved at tage udgangspunkt i midten af baggrunden, sikre at det vil give et godt indtryk på ethvert aspekt ratio. Dertil kan man sørge for at andre elementer på skærmen også justere sig selv efter kanterne af skærmen, for at sikre at de ikke bliver placeret på en måde der giver et upoleret indtryk [3].

Dette koncept benyttes også på andre platforme. F.eks. har det online kortspil Hearthstone grafik der går ud over selve den kvadratiske brugerflade, således at grafikken på en widescreen skærm også giver et godt visuelt indtryk, uden et behov for at benytte sig af letterboxes i siderne. På figur 2.8 og 2.7 ses det hvordan Heartstone tilpasser sig den givne skærms aspekt ratio.



Figure 2.7: Hearthstone med aspekt ratio 4:3



Figure 2.8: Hearthstone med aspekt ratio 16:9

Når man skal konstruerer et spil, skal det typisk kunne spilles på mange forskellige platformer med forskelligt hardware, og for at alle spillerne på de forskellige plaformer, skal kunne få den bedst mulig oplevelse til det hardware de har, bliver man nødt til at have forskellige skærmopløsninger. Når man har forskellige skærmopløsning i sit spil, kan der opstå en række problemer, som man som udvikler bliver nødt til at løse. Disse problemer kan summeres ned til tre: Skalering, forskellige aspekt ratio, og pixelkoordinater.

Bibliography

- [1] Apple. "What are Letterboxing, Pillarboxing and Windowboxing?" In: *apple* (2015). URL: <https://support.apple.com/en-us/HT204429>.
- [2] Jeff Atwood. "Widescreen and FOV". In: *codinghorror* (2007). URL: <http://blog.codinghorror.com/widescreen-and-fov/>.
- [3] Rubén Garat and Ariel Coppes. "Our solution to handle multiple screen sizes in android part one". In: *Gemserk Blog* (2013). URL: <http://blog.gemserk.com/2013/01/22/our-solution-to-handle-multiple-screen-sizes-in-android-part-one/>.
- [4] Andrew S. Gibson. "Aspect Ratio: What it is and Why it Matters". In: *digital-photography-school* (2013). URL: <http://digital-photography-school.com/aspect-ratio-what-it-is-and-why-it-matters/>.
- [5] Computer Hope. "Letterbox". In: *Computer Hope support* (x). URL: <http://www.computerhope.com/jargon/l/letterbo.htm>.
- [6] Steam. "Steam Hardware and Software Survey: February 2015". In: *steampowered* (2015). URL: <http://store.steampowered.com/hwsurvey>.
- [7] Wikipedia. "List of common resolutions". In: *wikipedia* (sidste revidering i 2015). URL: http://en.wikipedia.org/wiki/List_of_common_resolutions.
- [8] Caron Wills. "Older games with 4:3 aspect ratio are stretched in full screen". In: *codeweavers* (2014). URL: https://www.codeweavers.com/support/wiki/linux/faq/43_game_stretch.