Version 0.1 alpha

HTML5 2D Rendering Engine

Technical Design

Nederlandse versie

In samenwerking met

Thomas van der Berg

Inleiding

HTML komt met nieuwe mogelijkheden hun nieuwe Canvas en WebGL. Daarvoor moet er natuurlijk een engine komen die het ons makkelijker maakt om er iets voor te produceren. Hierbij kwam het idee om deze nieuwe rendering engine te bouwen en daarvoor is dit Technisch Design document ontstaan om nieuwe developers de basis uit te leggen hoe ze met deze rendering engine aan de slag kunnen of waar ze wat kunnen aanpassen.

Gezien we niet in dit hele document alle classes gaan uitleggen verwijzen we diegene die die informatie wel graag wil naar de ./docs/ map hierin vind u “index.htm” en kunt u alle classes doorzoeken plus extra informatie.

Deze engine genaamd “HTML5_2DRE” afkorting voor: “HTML5 2 Dimensional Rendering Engine” maakt alleen gebruik van de canvas mogelijkheid in HTML5.

Omdat de rendering engine nog steeds in de alpha fase is, Is nog niet alles als goed werkend bestempeld. De text rendering heeft zo zijn euvels met het switche van lettertype/grootte omdat deze continue het font opnieuw inlaad bij elke kleine verandering. Dit veroorzaakt lag.

Contents

Korte beschrijving

De render engine werkt vooramelijk door drawableObjects. Deze drawableObjects worden inherited van de BaseDrawable class. Hun update en draw functies en andere functies van hun children worden automatisch aangeroepen door de main loop. Spellen horen gemaakt te worden door objecten aan te maken die inheriten van de objecten in Model/HTML5_2DRE_Drawables.js, update functies voor deze objecten te maken en om deze objecten toe te voegen aan het spel door de addDrawable(*object*) functie.

Game of ander programma opstarten

Om jou spel of ander programma op te starten via deze rendering engine moet je ervoor zorgen dat dit programma na de engine wordt ingeladen. Doordat javascript het aanpassen van objecten altijd op elk moment toelaat kun je ervoor kiezen om 1 van de MVC objecten uit te breiden (extend) of je kunt ervoor kiezen geheel je eigen engine te bouwen. Hiervoor hoef je alleen de objecten die je gerenderd wil hebben aan de Model door te geven (Model.addDrawable(object, depth)) de rendering engine regelt de rest.

In dit document gaan we ervan uit dat er een spel wordt gemaakt.

Voor algemene update en draw calls kun je de View.draw en Model.update overschrijven. Overschrijf niet de View.base_draw of Model.base_update anders zal de engine stoppen met werken zoals die hoort.

Overschrijven of aanvullen van functies/velden:

```
View.extend({
        draw : function() {
                // your code
        },
        canvasID : “other name”
});
```

Instantieren / clonen:

Vaak moet de orginele classe waarvan we een nieuwe hebben gemaakt hetzelfde blijven anders veranderen we deze basis classe wat ten nadele van volgende objecten zal zijn, om dit te voorkomen is het toevoegen van de clone() functie voldoende.

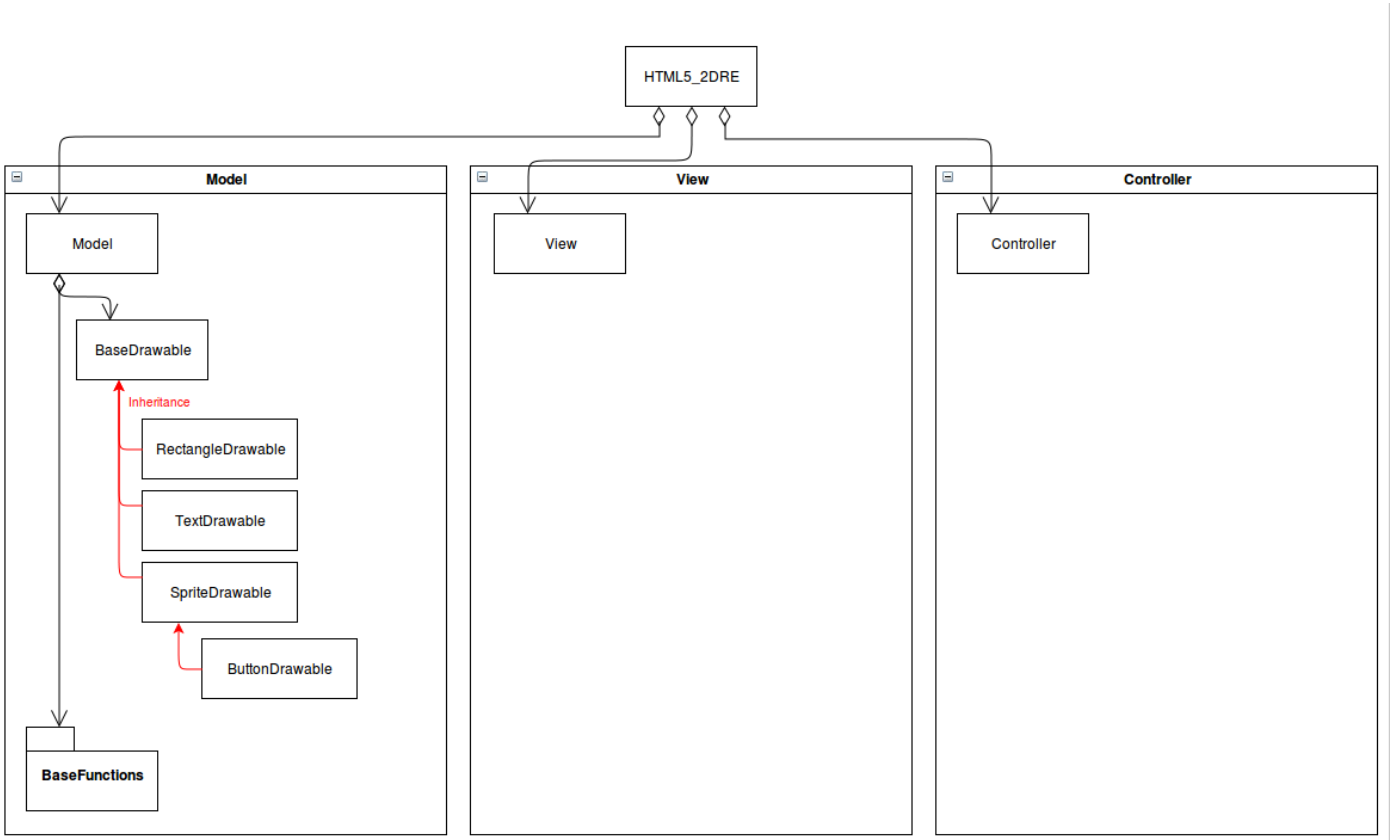
```
NewObject = BaseDrawable.clone();
```

Verhogen van de fps limit

Het spel kan natuurlijk ook sneller draaien, mocht dit gewilt zijn verander dan de View.fps waarde naar een hoger getal. Dit moet wel gedaan worden voordat het spel opstart.

Hiërarchisch diagram

Hier is het overzicht van welke classes er standaard in het spel zitten, deze kunnen uitgebreid worden of zo gelaten.

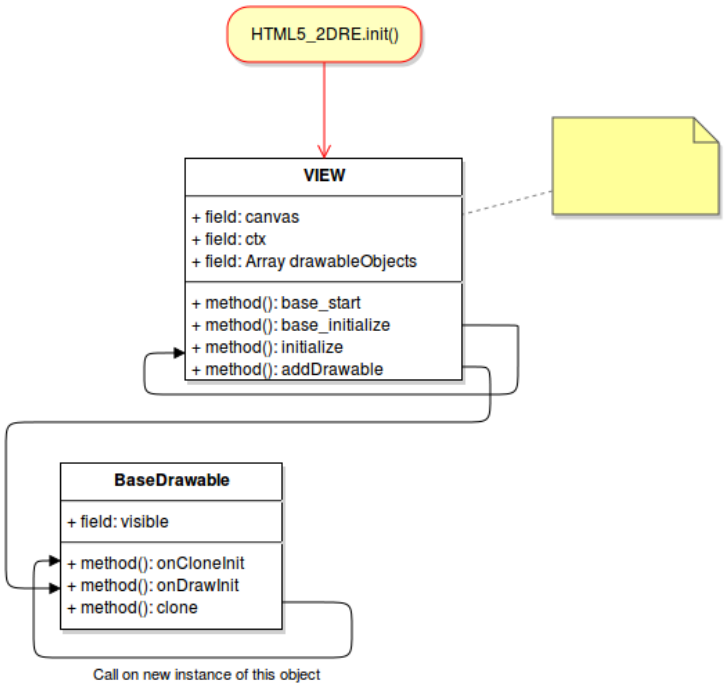


Startprocedure

De rendering engine(HTML5_2DRE.js) moet voor de game ingeladen worden in het HTML bestand. Hij start door extra functies en de standaard BaseDrawable objecten te laden. Daarna definieert hij alle functies die nodig zijn voor de main loop. De HTML5_2DRE_start functie wordt aangeroepen als alle documenten geladen zijn door middel van window.onload.

Hierna wordt het spelscript geladen. Als je het spel voor de engine inlaad dan zullen er errors ontstaan omdat het spel niet kan communiceren met de rendering engine.

Wanneer alles ingeladen is, wordt de HTML5_2DRE _start functie in HTML5_2DRE.js aangeroepen. Die laadt dan informatie over de HTML5 renderer in, en vraagt indirect de HTML5_2DRE_initialize functie aan als deze is gedefinieerd, hierna start de main loop.

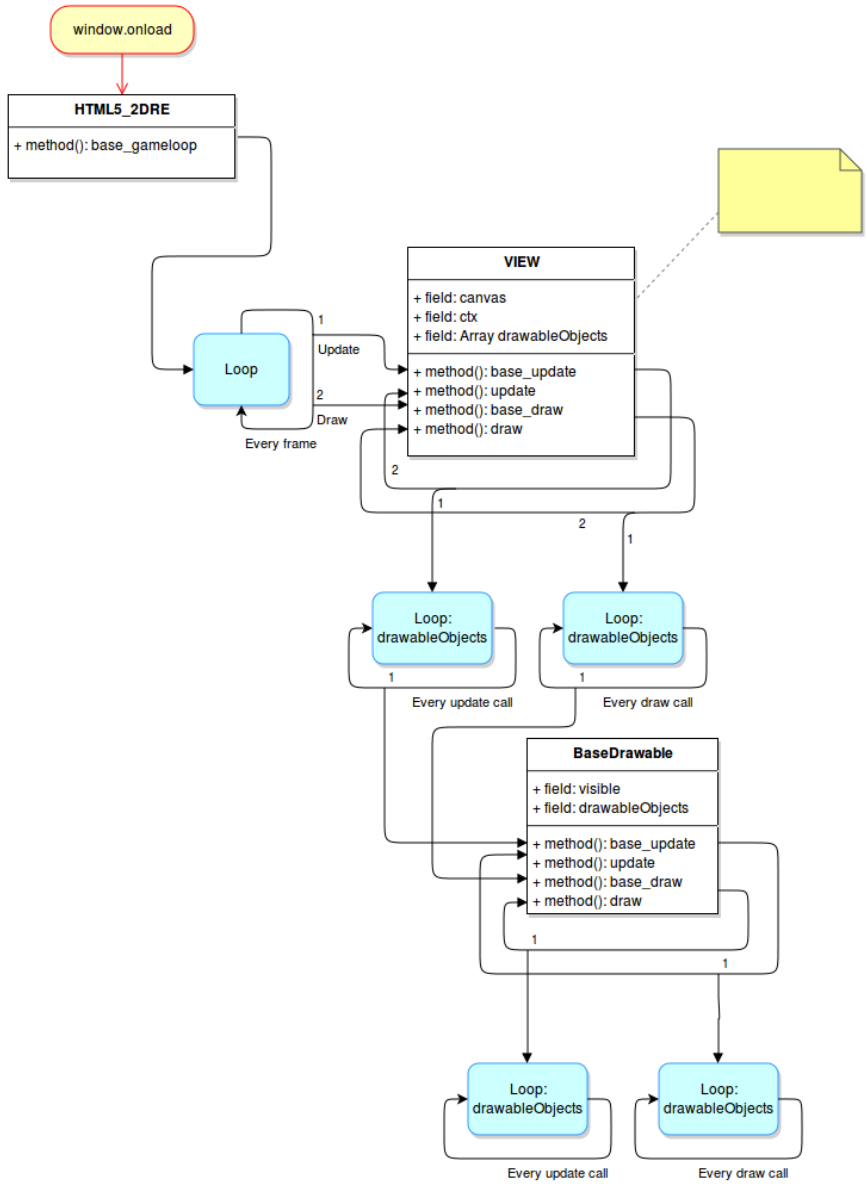


Main loop

De main loop wordt uitgevoegd door de functie HTML5_2DRE_gameLoop in HTML5_2DRE.js. Deze functie zorgt er eerst voor dat de tijd goed gezet wordt voor de deltaTime (de tijd die het heeft gekost van de vorige frame totaan deze, seconden), en voert dan de base_update functie uit, daarna de base_draw functie en ten slotte roept hij zichzelf op na de door ‘fpsinterval’ opgegeven tijd.

In base_update wordt elk object dat toegevoegd is aan de drawableObjects lijst geupdate, daarna wordt de FPS counter geupdate en ten slotte wordt de update functie van het spelscript uitgevoerd als het bestaat.

In base_draw wordt elk object in drawableObjects getekent, en ten slotte de draw functie van het spelscript uitgevoerd als het bestaat.



Drawable Objects en functionaliteiten

Er zijn standaard 4 BaseDrawable objecten die gebruikt kunnen worden om eigen specifieke objecten te maken. Hier is een overzicht van deze classen en wat ze doen:

RectangleDrawable – Tekent een blokje met een opgegeven kleur, kan ook een border bevatten.

TextDrawable – Schrijft tekst, kan ook een border bevatten.

SpriteDrawable – Renderd een afbeelding.

ButtonDrawable – Zelfde als SpriteDrawable maar dan met met gefocust op klik mogelijkheden en veranderd de cursor.

Alle BaseDrawables hebben standaard functies die door de View worden behandeld, hierin kan een eigen functie gezet worden zodat deze wordt aangeroepen. Hier volgt de standaard functies die deze engine behandeld:

Onclick – Als er op het object geklikt wordt (laat alleen bij Buttons een andere cursor zien)

Onhover – Als we over het object heen gaan met de muis.

Onmouseout – Als we met de muis van het object af gaan.

Eindprocedure

Deze rendering engine heeft geen procedure ter afsluiting, dit omdat er verder niks hoeft afgesloten te worden. Javascript is er op gebouwd dat het eenmalig initieert en functies aanroept en uit het niets weer alle scripts weg doet.