Het aansturen van render-engines

Een onderzoek naar het aansturen van meerdere televisie render-engines met één applicatie

Tygo Koopman
UNITED B.V. | SAXION HOGESCHOOL

Informatie Pagina

Afstudeerproject sep 2020 - feb 2021

Klant: United B.V.

Bedrijfsbegeleider: Gerard de Vries

Afstudeerdocent: Mark Schipper

Student: Tygo Koopman

Studentnummer: 431582

Studie: Creative Media and Game Technologies (KNT)

Email: tygokoop@gmail.com

Datum: 12-01-2020

Samenvatting

Broadcast bedrijven maken gebruik van gespecialiseerde hard- en software om tijdens live televisieprogramma's grafische vormgeving te kunnen genereren en af te spelen. Wanneer deze bedrijven graphics uitspelen met meerdere render-engines, of wanneer er wordt samengewerkt met bedrijven die andere engines gebruiken, moet de aansturing van deze graphics per engine opnieuw geprogrammeerd worden. In deze scriptie wordt onderzocht hoe verschillende render-engines aangestuurd kunnen worden door één applicatie.

Het doel van dit onderzoek is om erachter te komen hoe een uitbreiding zou kunnen worden gemaakt op Graphics Connect. Hiervoor is de volgende onderzoeksvraag opgesteld: 'Hoe kan een systeem ontworpen worden dat meerdere live graphics render-engines kan aansturen zodat de gebruiker geen nieuwe software hoeft te programmeren bij gebruik van dezelfde content?'

Om tot een goed resultaat te komen is onderzocht hoe vormgevingen te maken zijn in XPression. Vervolgens is onderzocht hoe Graphics Connect communicatie kan ontvangen en versturen naar render-engines en aansturingssoftware voor de graphics. Om hierachter te komen is veel gebruik gemaakt van experts die veel ervaring hebben met deze softwares. Ook is er literatuuronderzoek uitgevoerd en zijn er prototypes gemaakt.

Uit deze onderzoeken is gebleken dat XPression aangestuurd kan worden door hier direct commando's heen te sturen, maar dat dit ook kan als er uitbreidingen worden gemaakt in Graphics Connect. Voor deze uitbreidingen moeten in de libraries, die Graphics Connect gebruikt, XPression als servertype worden toegevoegd. Ook moet in Graphics Connect per functie een uitbreiding komen voor XPression.

Als vervolg van deze scriptie is het mogelijk om te onderzoeken of de verschillende engines ook tegelijkertijd aangestuurd zouden kunnen worden. Hierdoor zou een gebruiker niet meer hoeven wisselen tussen de verschillende servertypes.

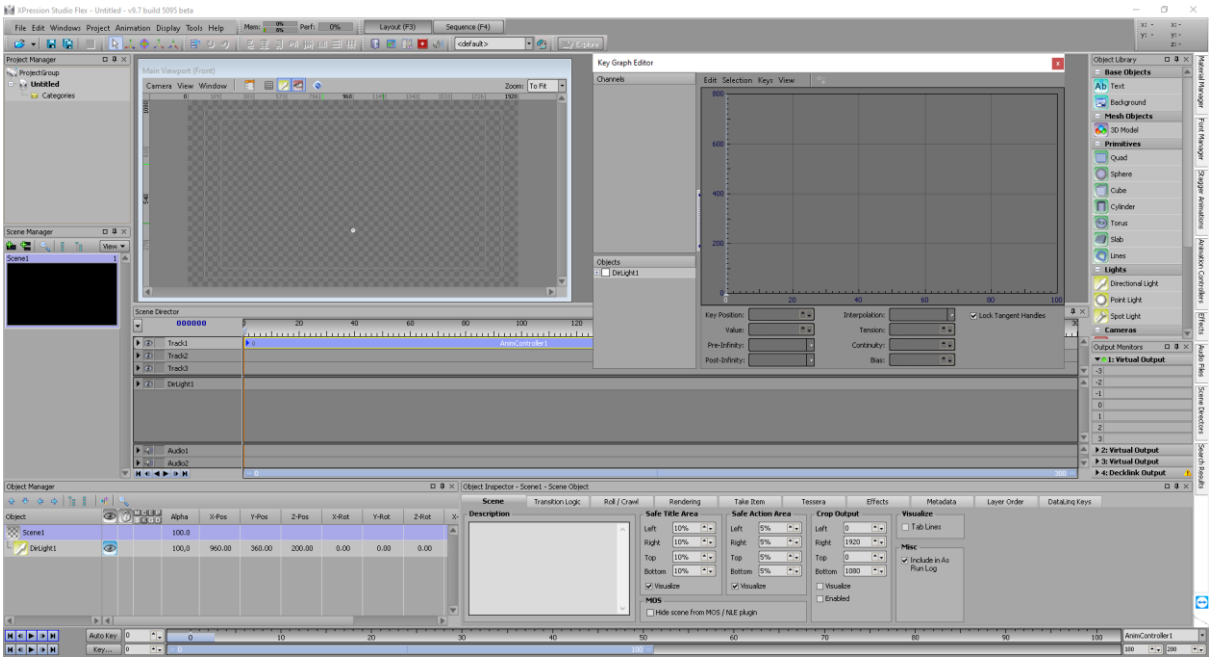
Inhoudsopgave

1. Introductie

1.1. United en graphics

United is een groot Nederlands broadcast bedrijf wat gespecialiseerd is in live televisieproducties. Voor live televisieprogramma's zoals sportevenementen of spelshows worden live graphics (Walker, 2001, p. 2) gebruikt. Dit zijn bijvoorbeeld klokjes, scores of rode kaarten (zie figuur 1). Deze graphics kunnen worden afgespeeld door bepaalde render-engines (of graphics-engines) en vervolgens live over het beeld heen gezet. Deze engines kunnen animaties of video's afspelen en data, zoals tekst of afbeeldingen, aanpassen. Voor broadcast bedrijven bestaan tientallen engines van gratis CasparCG (*CasparCG*, z.d.) tot de zeer uitgebreide XPression (*XPression*, z.d.) (zie figuur 2).

***Figuur 1**: Een voorbeeld van broadcast graphics (Avid, z.d.)*



***Figuur 2**: XPression in gebruik*

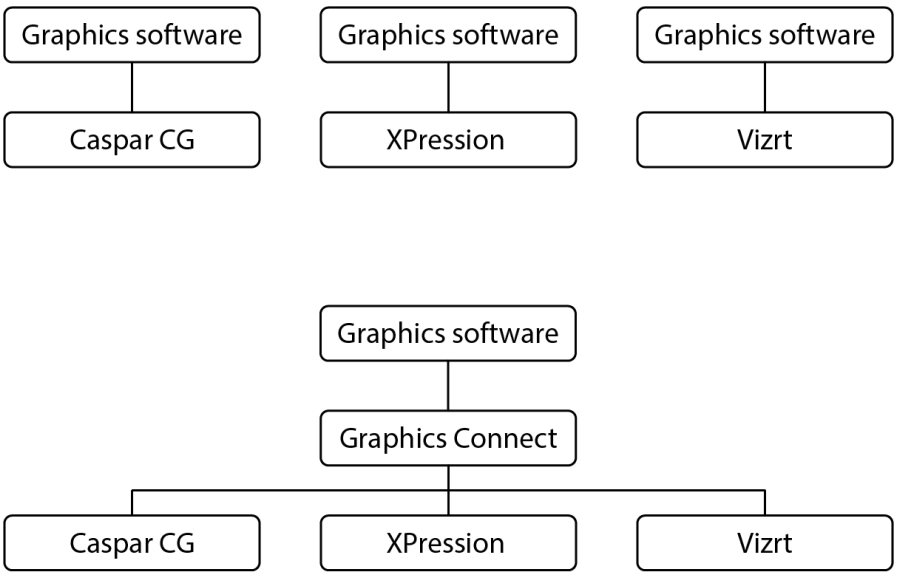
In render-engines kun je naast animatie en templates maken, dit ook uitspelen en genereren als videobron. Voor ingewikkeldere televisieprogramma's, zoals spelshows of sportwedstrijden heeft United eigen software ontwikkeld welke animaties genereerd en afspeelt. Deze software is geprogrammeerd met de SDK of Library van de render-engine zelf, zodat ze live teksten kunnen aanpassen. SDK staat voor Software Development Kit, dit softwarepakket bevat verschillende

applicaties met bijbehorende libraries en nodige ondersteuning (Valdellon, 2020). De United programmeurs gebruiken deze SDK voor het bouwen van eigen applicaties.

1.2. Euro Media Group

United is onderdeel van de Europese broadcastgroep Euro Media Group (*Euro Media Group*, z.d.), waar ook het Belgische Videohouse deel van uitmaakt. United en Videohouse willen nauwer gaan samenwerken, startende bij de grafische afdeling. Videohouse maakt geen gebruik van XPression, maar van Vizrt (*Vizrt*, z.d.). Deze engine wordt met een andere SDK aangestuurd dan XPression. Uit een gesprek met Gerard de Vries (Senior Software Engineer, United) bleek dat voor eenzelfde soort televisieprogramma de software en de templates in de render-engine opnieuw gemaakt moet worden. Volgens Gerard gaat in de software van bijvoorbeeld 'Max pubquiz' 8 dagen werk zitten. De tijd die gebruikt wordt voor het maken van die software niet nodig, als je de bestaande software niet opnieuw zou hoeven programmeren.

Videohouse maakt gebruik van Vizrt voor de meeste producties, maar voor een aantal kleinere projecten gebruiken zij ook CasparCG. Videohouse heeft een programma ontwikkeld die commando's kan ontvangen en kan sturen naar CasparCG of naar Vizrt. Dit programma heet Graphics Connect en zorgt ervoor dat ongeacht de manier van uitspelen je maar één keer een besturingsprogramma moet maken (zie figuur 3).



***Figuur 3**: De huidige manier van software programmeren (boven) tegenover een vernieuwde manier van software programmeren door middel van een vertaalscript (onder)*

Graphics Connect is te vergelijken met een vertaalprogramma, zoals Google translate. Iets soortgelijks doet Graphics Connect ook, alleen dan met commando's die de verschillende render-engines aanspreken. Graphics Connect hoeft maar 1 keer gemaakt te worden, waardoor United bestaande graphics software van Videohouse direct kan gebruiken of andersom. Het huidige plan is dat er een uitbreiding komt voor de bestaande Graphics Connect, waarmee United ook XPression kunnen aansturen.

1.3. Doel van de klant

De collega's van Videohouse gebruiken een andere manier van uitspelen van live graphics. Op dit moment moet United opnieuw de graphics maken en programmeren als een programma van België naar Nederland komt. Uit het gesprek met Gerard de Vries is gebleken dat vooral het programmeren van de software zeer veel tijd kost. Om al deze extra moeite in te perken, is er aan het einde van dit onderzoek een programma die de aansturing van het graphics pakket om kan vormen naar commando's om verschillende render-engines aan te kunnen sturen. Het doel voor United is dit programma te gaan gebruiken bij het ontwikkelen van nieuwe televisieproducties, zodat ze veel tijd kunnen besparen als United meer gaat samenwerken met Videohouse.

Een bijkomend voordeel van dit onderzoek is dat Videohouse en United slechts een uitbreiding hoeven te schrijven op het bestaande Graphics Connect, indien ze meer gaan samen werken met andere bedrijven uit de Euro Media Group. Hiermee is het een extra stap in de richting van de nauwere samenwerkingen tussen deze verschillende bedrijven.

2. Probleemstelling

United besteedt onnodig tijd aan het programmeren van graphics software voor televisieproducties. Als deze televisieproducties bij zusterbedrijf Videohouse al bestaan, kan United de bestaande software hergebruiken waardoor veel kostbare tijd bespaard wordt. De manuren die hiermee bespaard worden kunnen vervolge elders worden ingezet.

3. Formulering van de hoofdvraag en subvragen

3.1. Hoofdvraag

Hoe kan een systeem ontworpen worden dat meerdere live graphics render-engines kan aansturen zodat de gebruiker geen nieuwe software hoeft te programmeren bij gebruik van dezelfde content?

3.2. Subvragen

- Hoe kan in XPression het visuele vormgevingspakket (graphics-pakket) van een televisieproductie ontworpen en geanimeerd worden?
- Hoe kan een besturingssysteem geprogrammeerd worden die de graphics van een televisieproductie aanstuurt?
- Hoe kan door middel van Graphics Connect, communicatie die met verschillende render-engines te gebruiken is, verstuurd worden?
- Hoe kan een bestaand vormgevingspakket en software communiceren via Graphics Connect, zodat het ook gebruikt zou kunnen worden met andere protocollen?

4. Onderzoeksmethoden

Als extra informatie voor de onderzoeksmethoden zijn CMDmethods.nl (HAN - University of Applied Sciences & AUAS - Amsterdam University of Applied Sciences, z.d.) en ictresearchmethods.nl (bonestroo, z.d.) gebruikt.

4.1. Hoe kan in XPression het visuele vormgevingspakket (graphics-pakket) van een televisieproductie ontworpen en geanimeerd worden?

Om een goed antwoord te kunnen geven op de vraag hoe een vormgevingspakket ontworpen en geanimeerd kan worden, is er gekozen om een bestaand vormgevingspakket na te maken. Door het namaken van een bestaand vormgevingspakket wordt het duidelijk hoe XPression werkt en kan er getest worden met het aansturen van deze vormgeving. Het maken van deze vormgeving en het programmeren van de aansturing hiervan gaan daarom ook hand in hand.

Onderzoeksmethoden: Day in the life, tinkering, prototyping

Bronnen die gebruikt worden: Experts, websites, een video van een televisieproductie met graphics, XPression helpfile

4.2. Hoe kan een besturingssysteem geprogrammeerd worden die de graphics van een televisieproductie aanstuurt?

De programmatuur van de aansturing van het vormgevingspakket zal worden gemaakt in C#. Over C# is veel te vinden op het internet, dus voor het programmeren zal veel gebruik gemaakt worden van bronnen op het internet. De implementatie van de XPressionSDK is genoteerd in een helpfile die geleverd wordt bij de aanschaf van XPression (figuur 9).

Onderzoeksmethoden: Tinkering, day in the life, Prototyping

Bronnen die gebruikt worden: Websites, Fora, XPression helpfile, experts

4.3. Hoe kan door middel van Graphics Connect, communicatie die met verschillende render-engines te gebruiken is, verstuurd worden?

Videohouse heeft een programma geschreven die beide CasparCG en Vizrt kan aansturen. In de zogenoemde Graphics Connect zal worden onderzocht hoe deze functies kan aanroepen in de modules van CasparCG en Vizrt. Ook zal worden gekeken naar waar er onderscheid wordt gemaakt tussen beide onderdelen.

Onderzoeksmethoden: Available product analysis, Literature Study

Bronnen die gebruikt worden: Experts van Videohouse, Github en readme file van Graphics Connect, websites

4.4. Hoe kan een bestaand vormgevingspakket en software communiceren via Graphics Connect, zodat het ook gebruikt zou kunnen worden met andere protocollen?

De vormgeving die is gemaakt in XPression kan worden aangestuurd door de software die is geprogrammeerd in C#, door middel van de XPressionSDK. Er wordt onderzocht of deze SDK te implementeren is in Graphics Connect. Vervolgens zal er worden gekeken of de functies binnen Graphics Connect aan te sturen zijn in een losstaande applicatie.

Onderzoeksmethoden: Prototyping

Bronnen die gebruikt worden: Github en readme file van Graphics Connect, fora, websites

5. Scope

In overleg met de klant is besloten dat er alleen onderzoek wordt gedaan naar hoe Graphics Connect werkt en hoe die uitgebreid kan worden om te werken met XPression. Er zal worden gekeken naar één van de functies van XPression en deze zal worden uitgewerkt. Niet alle functies die in de XPressionSDK zitten zullen worden gemaakt. Als duidelijk wordt hoe er functies kunnen worden geschreven in Graphics Connect, kan de klant dit uitbreiden tot alle functies die ze nodig heeft.

6. Resultaten

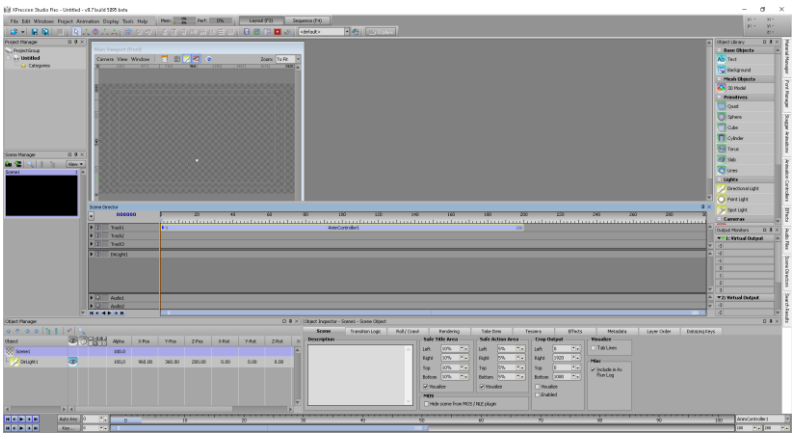
6.1 Hoe kan in XPression het visuele vormgevingspakket (graphics-pakket) van een televisieproductie ontworpen en geanimeerd worden?

Day in the life

XPression is software dat is gemaakt voor live graphics bij televisie-uitzendingen. Omdat dit een zeer specifiek softwarepakket is, is er weinig documentatie over. Dit is ook de voornaamste reden om met collega's mee te kijken terwijl zij bezig zijn met het maken van animaties of vormgeving in XPression.

Het eerste wat er normaal gesproken gebeurt als er een nieuw graphics-pakket wordt ontwikkeld, is het aanmaken van een nieuw project in XPression. Hierin wordt een standaard naamgeving gebruikt om overzicht te houden voor andere collega's als deze in hetzelfde bestand moeten werken. Deze naamgeving is uitgelegd, zodat andere projecten ook duidelijk te begrijpen zijn.

Aan het einde van een dag meekijken met Gerard de Vries, heeft hij ook de benodigde knoppen en functies uitgelegd om de basis van XPression te begrijpen (figuur 4). Uit dit gesprek bleek ook dat het handig was om een al bestaande vormgeving van een televisieprogramma na te maken. Dit heeft hij aangeraden, omdat hij van mening is dat iemand die al enige ervaring met animatie en grafische ontwerp software heeft, het meest en het makkelijkst deze software kan leren begrijpen door er zelf rustig doorheen te kijken en zelf een vormgeving te maken.



***Figuur 4**: Een nieuw XPression bestand*

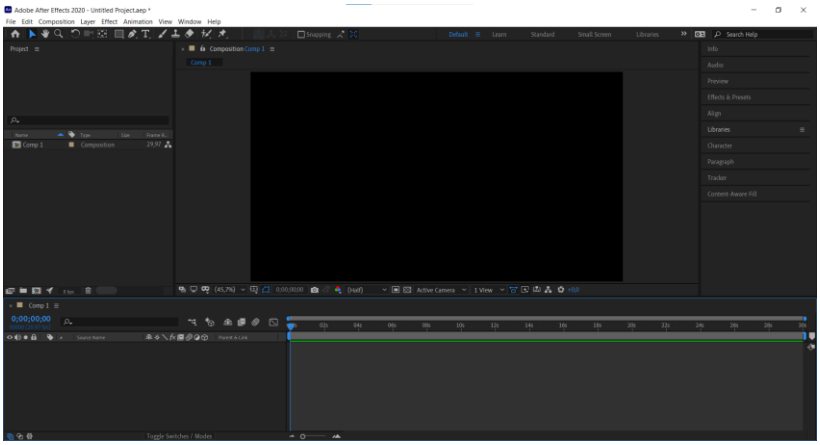
Tinkering

Zoals eerder genoemd heeft Gerard de Vries aangeraden om het vormgevingspakket van een bestaand televisieprogramma na te maken om de software te leren kennen. Deze software zou eventueel later gebruikt kunnen worden voor het testen van geïmplementeerde funcites. In overleg is besloten om het zes-letter spel en de draaiende ballen van de oude vormgeving van Lingo na te maken (Npostart, 2011, 03:23-04:03). Lingo is een Nederlands spelprogramma, gemaakt door IDTV, waarbij deelnemers woorden moeten raden om zo ballen te mogen pakken en hiermee geld te verdienen (Lingo, 2020). Lingo is gekozen als vormgeving om na te maken, omdat hier een aantal lastige animaties in zitten en de aansturing van een spelshow de limitaties en uitdagingen van de besturingssoftware laten zien (figuur 5).

De eerste fase van het maken van Lingo is om de XPression software te leren kennen. Dit wordt gedaan door tools en vensters, die bekend voorkomen uit andere softwares, te proberen in

XPression. Zoals te zien in figuur 6 zijn elementen zoals de tijdlijn, een objecten venster en een viewport in XPression te vergelijken met de After Effects software van Adobe (Adobe, z.d.). Door deze tools en vensters te testen werden deze tools ook binnen XPression duidelijk.

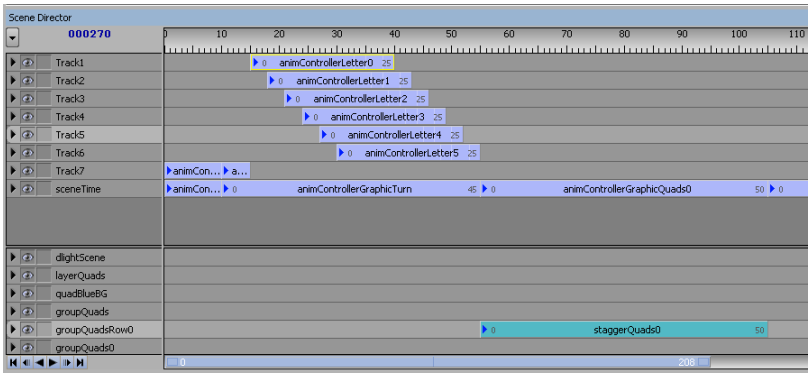
***Figuur 5**: Het ballen onderdeel (links) en het letterspel (rechts) van Lingo (Npostart, 2011, 03:24-04:01)*



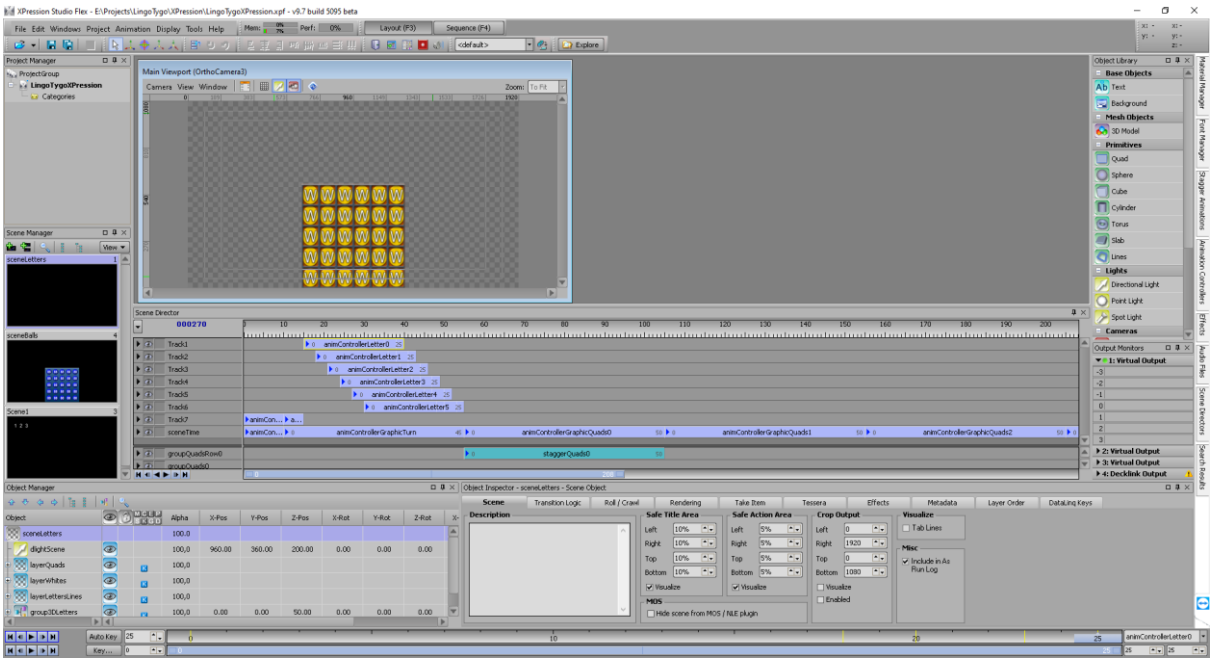
***Figuur 6**: Een leeg After Effects bestand*

Prototyping

In XPression kun je ‘stagger animations’ maken. Dit zijn animaties die per karakter of per object in een groep herhaaldelijk uitvoeren (Ross Video, 2017, 00:08-00:27). In het geval van de letters in de 3D omdraai-animatie van de eerste zes letters in het Lingo spel, is dit uiterst handig. Op deze manier hoeft de animator niet per letter een keyframe te zetten. Een keyframe is een begin-, tussen- of eindpunt van een animatie (Wikipedia contributors, 2020). Deze ‘stagger animations’ zijn dus voor de eerste zes letters van Lingo gebruikt, maar deze zijn niet overal handig. Dit komt doordat de animaties niet altijd allemaal precies hetzelfde zijn en dan zijn stagger animations onpraktisch (figuur 7). In figuur 8 is te zien hoe het Lingo project er uit ziet nadat alle objecten aanstaan en de animaties geïmplementeerd zijn.



***Figuur 7**: Stagger animations vergeleken met normale animaties*

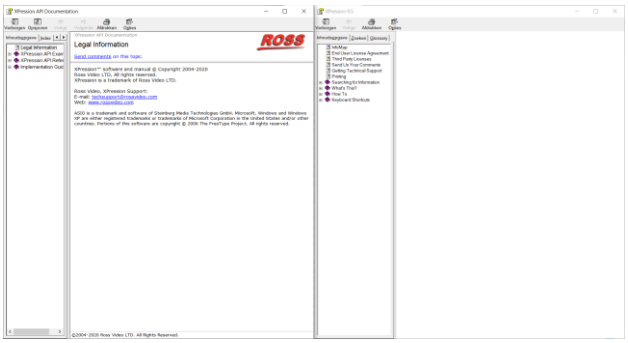


***Figuur 8**: Het maken van Lingo in XPression*

6.2 Hoe kan een besturingssysteem geprogrammeerd worden die de graphics van een televisieproductie aanstuurt?

Tinkering

Om te begrijpen hoe XPression functies kan uitvoeren, zijn enkele voorbeelden van de XPressionSDK bestudeerd (figuur 9). Hierin staan voorbeelden omschreven hoe een gebruiker kan beginnen met het programmeren van zijn eigen aansturingssoftware (Bijlage A). In voorbeeld ‘Loading a project’ staat beschreven hoe een applicatie de XPressionSDK moet inladen en hoe vervolgens verbinding gemaakt kan worden met XPression.



***Figuur 9**: De helpfiles van XPression en XPressionSDK*

De voorbeelden van de helpfile zijn geschreven in de programmeertaal ‘Visual Basic’. Visual basic is een programmeertaal die redelijk vergelijkbaar is met C# (Visual Basic, 2020) en is daarom te begrijpen door iemand die C# kent. De XPressionSDK helpfile is gescheiden in ‘Examples’, ‘Objects’, ‘Methods’, ‘Properties’, ‘Events’ en ‘Enumerations’. De code en toepassingen van de meeste onderdelen van XPression zijn kort uitgelegd in dit bestand. Door deze door te lezen en verschillende functies te proberen in een eigen applicatie, is duidelijk geworden hoe verschillende animaties te starten zijn, data aan te passen is en hoe andere scenes te laden zijn.

Day in the life

Tijdens een dag meekijken met de uitvoering van live graphics bij een televisie-uitzending, is opgevallen dat de gebruikers van de besturingssoftware een grafische interface gebruiken (Microsoft, 2019). Tijdens een uitzending is er namelijk niet altijd veel tijd om de juiste graphics te maken en te sturen naar Xpression. De text- en checkboxen zijn dus bedacht om tijd te besparen en minder fouten te maken in animaties en data, aldus Bryan Boor van United.

Prototyping

De vormgeving van figuur 5 geeft direct na het invullen van een woord aan of de ingevulde letters op de juiste plaats staan en als dit niet het geval is of ze dan onderdeel uitmaken van het woord op een andere plek (figuur 10). Voor alle letters moeten beide situaties gecheckt worden. Door dit te programmeren kan de juiste achtergrond per letter door gestuurd worden naar XPression (figuur 11).

***Figuur 10**: Lingo graphics zoals het er uit ziet op televisie*

```
void checkWord()
{
    char[] lingoWordCharArray = new char[lingoWordLength];
    char[] txtGuessedWordCharArray = new char[lingoWordLength];

    lingoWordCharArray = lingoWord.ToUpper().ToCharArray();
    txtGuessedWordCharArray = txtGuessedWord.Text.ToUpper().ToCharArray();

    for (int i = 0; i < lingoWordCharArray.Length; i++)
    {
        if (lingoWordCharArray[i] == txtGuessedWordCharArray[i])
        {
            lingoWordCharArray[i] = "*".ToCharArray()[0];
            txtGuessedWordCharArray[i] = "+".ToCharArray()[0];

            redSquareQuad[i + (lingoWordLength * lingoWordRow)].Visible = true;
        }
    }

    for (int i = 0; i < lingoWordCharArray.Length; i++)
    {
        for (int j = 0; j < txtGuessedWordCharArray.Length; ++j)
        {
            if (lingoWordCharArray[i] == txtGuessedWordCharArray[j])
            {
                lingoWordCharArray[i] = "-".ToCharArray()[0];
                txtGuessedWordCharArray[j] = "/".ToCharArray()[0];

                yellowCircleQuad[j + (lingoWordLength * lingoWordRow)].Visible = true;
            }
        }
    }
```

***Figuur 11**: Code die de ingevulde letters kan controleren op juiste plek en juiste volgorde*

In Lingo mogen de deelnemers van het spel, nadat ze een woord goed hebben geraden, een bal grabbelen met een getal daarop. Als dit getal wordt opgelezen door de deelnemer, ziet de kijker in beeld een blauw balletje met het corresponderende getal geel worden, om te markeren dat dit getal al is geweest. Door middel van een checkboxen matrix in de software, kan de bestuurder van de software tijdens de uitzending een nummer selecteren en vervolgens met een knop de animatie laten spelen.

De bovengenoemde functies zitten verwerkt in een visuele software zoals te zien is in figuur 12. Hierdoor kunnen de eindgebruikers van de software gemakkelijk en snel de juiste informatie doorsturen naar XPression.

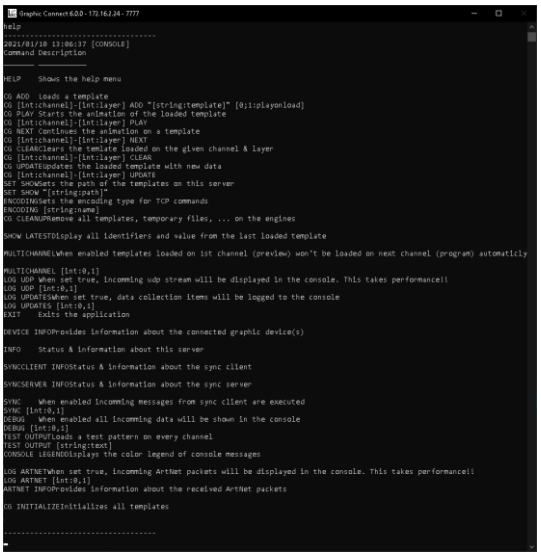
***Figuur 12**: De aansturingssoftware voor het besturen van Lingo graphics*

6.3 Hoe kan door middel van Graphics Connect, communicatie die met verschillende render-engines te gebruiken is, verstuurd worden?

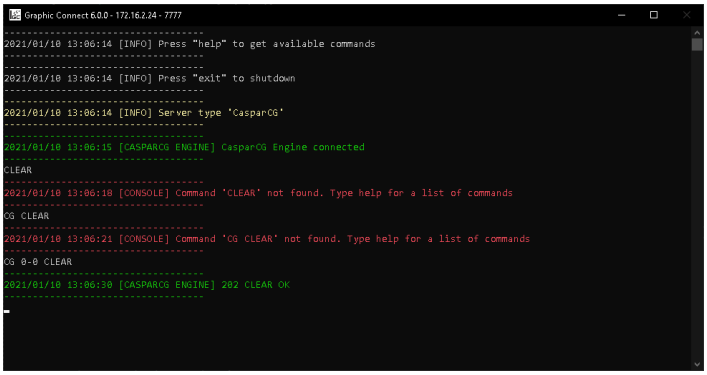
Available product analysis

Zoals eerder verteld maakt Videohouse gebruik van de render-engines Vizrt en CasparCG. Hiervoor hebben zij een eigen applicatie ontwikkeld, Graphics Connect, die beide engines kan aansturen. Het grootste deel van Graphics Connect is gemaakt door Videohouse. Om XPression toe te kunnen voegen aan Graphics Connect moet eerst duidelijk worden hoe Graphics Connect omgaat met de huidige geïmplementeerde engines. United heeft Videohouse verzocht om de source code te krijgen van Graphics Connect, zodat de student goed kan bestuderen hoe Graphics Connect is opgebouwd (Videohouse, 2020)[1].

Graphics Connect bestaat uit verschillende onderdelen, namelijk een configuratie bestand waar de gebruiker de instellingen en vooral het type engine kan selecteren, de code van de server zelf en codes per geïmplementeerde functies. De ‘Clear’ functie kan op het moment dat er content op het scherm te zien is een kanaal leeg maken. Deze functies zal uitgewerkt en uitgebreid in Graphics Connect. In figuur 13 is de helppagina van Caspar Connect laten te zien, als de gebruiker ‘help’ typt in de server. Met de notatie ‘CG 0-9 CLEAR’ wordt laag 10 op kanaal 1 weggehaald (figuur 14).



***Figuur 13**: De help pagina in Graphics Connect*

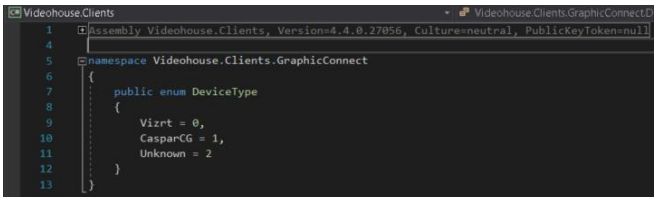


***Figuur 14**: Een onjuist en een juist ‘Clear’ Commando in Graphics Connect*

[1] Bron afkomstig van de Github (niet publiekelijk toegankelijk) van Videohouse.

Literature Study

Graphics Connect maakt gebruik van 'Dynamic-link libraries' of dll bestanden verkort. Dll bestanden zijn bibliotheken met functies die gebruikt kunnen worden door verschillende applicaties (Dynamic-link library, 2016). Deze bestanden kun je alleen maar lezen en niet herschrijven (Wikipedia contributors, 2020). Een aantal van de benodigde dll's worden ook door Videohouse gemaakt om functies en onderdelen van de graphics engines te kunnen gebruiken in Graphics Connect. Om de huidige functionaliteit van Graphics Connect uit te breiden, zodat ook XPression hiermee zou kunnen werken, zou in de Clients dll van Videohouse de enum 'DeviceType' moeten worden uitgebreid met de waarde van XPression (figuur 15). Een enum een variabele met een vast aantal waarden die namen hebben (Wikipedia contributors, 2020). Dit zou nodig zijn omdat bij elke functie van de render-engines een switch case zit met de verschillende server types (figuur 16).



Figuur 15*: Een functie van de Videohouse dll*

```
switch (_server.Type)
{
    case Videohouse.Clients.GraphicConnect.DeviceType.Vizrt:
        VizEngineClient VizClient = _server.GraphicDevices[channel] as VizEngineClient;
        VizEngineResponse VizResponse = null;

        // execute command
        VizClient.Layers[layer].Clear(out VizResponse);

        // get response
        response = VizResponse.Response;

        break;

    case Videohouse.Clients.GraphicConnect.DeviceType.CasparCG:

        xpEngine engine = new xpEngine();

        engine.ClearFrameBufferLayer(channel, layer);



        CasparCGClient casparClient = _server.GraphicDevices[0] as CasparCGClient;

        // execute command
        casparClient.CGClear(channel + 1, layer + 1, out CasparCGResponse CaspResponse);

        // get response
        response = CaspResponse.Response;
        break;
```

Figuur 16*: De 'Clear' functie van Graphics Connect*

6.4 Hoe kan een bestaand vormgevingspakket en software communiceren via Graphics Connect, zodat het ook gebruikt zou kunnen worden met andere protocollen?

Prototyping

Graphics Connect is geprogrammeerd als een server, dit wil zeggen dat je kunt verbinden met Graphics Connect door middel van een IP-adres met een specifieke poort (Microsoft, z.d.). Hetgeen echter niet beschreven in de documentatie van Graphics Connect. In eerste instantie was geprobeerd om verbinding te maken met de Graphics Connect lokale server door middel van 'TcpClient.Connect'. Volgens de documentatie van Mircosoft over '.Net', is te volgen hoe te verbinden met een netwerkserver via TCP (Mircosoft, z.d.). Dit blijkt niet de manier om te verbinden met Graphics Connect.

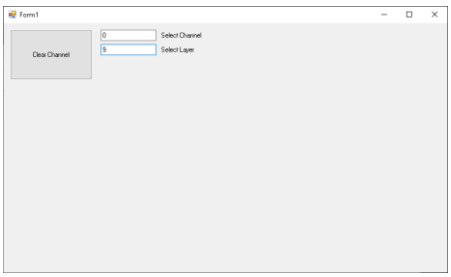
Videohouse verstrekte de originele code van een testprogramma wat gebruik maakt van Graphics Connect (Bijlage B). Hierin wordt gebruik gemaakt van een connectie functie die is geschreven in de Videohouse Graphics Connect dynamic-link library of 'Videohouse.Clients.dll'. Hiermee krijgt een gebruiker toegang tot de functies van Graphics Connect. Een van deze functies is het kunnen verbinden met Graphics Connect, zoals te zien is in figuur 16.

```
namespace GraphicsConnectTest
{
    public partial class Form1 : Form
    {
        GraphicConnectClient graphicConnectClient = new GraphicConnectClient();

        public Form1()
        {
            InitializeComponent();
            graphicConnectClient.Connect("127.0.0.1", 7777);
        }

        private void Btn_Clear_Click(object sender, EventArgs e)
        {
            graphicConnectClient.CGClear(Int32.Parse(TB_Channel.Text), Int32.Parse(TB_Layer.Text));

        }
    }
}
```

***Figuur 16**: Code van verbinden met Graphics Connect*

De 'Videohouse.Clients.dll' library geeft in elk ander programma dus toegang tot de functies die geschreven zijn in Graphics Connect. Na de uitgebreidere 'Clear' functie van Graphics Connect zou met de standaard oproep van Clear in de server ook het kanaal met de laag in XPression leeg moeten halen. Videohouse geeft aan dat de functies aan te roepen zijn door de naam van de functie zoals deze beschreven is in de code van Graphics Connect te gebruiken bij de client (figuur 16).

Als mogelijke implementatie in een software, is er een visueel 'Windows Form' (Microsoft, 2019) gemaakt met een knop die op een variabele laag in een variabele channel (figuur 17). Wanneer de eindgebruiker deze knop indrukt dan worden de lagen van de CasparCG en XPression Channels gecleared.



***Figuur 17**: Mogelijke implementatie van een clear knop*

7. Conclusie

In dit onderzoek is getracht de volgende vraag te beantwoorden: ‘Hoe kan een systeem ontworpen worden dat meerdere live graphics render-engines kan aansturen zodat de gebruiker geen nieuwe software hoeft te programmeren bij gebruik van dezelfde content?’. Voor deze vraag is onderzoek gedaan naar de huidige manier het maken van software graphics en naar Graphics Connect, een software van het zusterbedrijf Videohouse.

In XPression moet er goed worden gekeken naar de vormgeving die de klant wil. De onderdelen van een animatie of titel moeten worden opgebouwd uit kleinere objecten. Een gebruiker kan materialen met afbeeldingen toevoegen zodat daaroverheen de data kan komen te staan. Hiermee kunnen de objecten onafhankelijk of samen met elkaar worden geanimeerd om bewegingen in de vormgeving te maken.

In gesprek met collega's, die deze softwaresystemen tijdens televisie-uitzendingen moeten bedienen, is een duidelijk beeld geschept hoe deze besturingssystemen er uit moeten zien. Dit zijn grafische besturingspanelen met knoppen, tekstvelden en vinkjes. Hierdoor worden minder fouten gemaakt en kunnen live aanpassingen worden gemaakt aan animaties of data. Verder moet de code van deze software juiste data en scenes versturen wanneer er op verschillende knoppen wordt gedrukt of wanneer er tekst wordt ingevuld.

Om werkende commando’s of functies van render-engines uit te voeren moet Graphics Connect verbinding maken met de corresponderende servers of SDK’s van de render-engines. Als Graphics Connect deze verbinding heeft gemaakt kunnen er in de server van Graphics Connect functies worden uitgevoerd die worden vertaald naar de correcte functies per render-engine.

Bestaande softwarepakketten die vormgevingen aansturen, moeten worden aangepast, omdat de huidige softwares direct in verbinding zijn met de servers of SDK’s van de render-engines. De functies kunnen worden herschreven naar de functies van Graphics Connect. Functies die in Graphics Connect staan, doen hetzelfde als de directe commando’s, maar kunnen ook worden gebruikt door verschillende render-engines. Het maken van nieuwe software gaat op dezelfde manier als werknemers dit eerder ook al deden, alleen zijn de functies en de library aangepast naar Graphics Connect functies en libraries.

Uit dit onderzoek is gebleken dat Graphics Connect goed kan werken in combinatie met meerdere graphics render-engines. Doordat alle benodigde libraries voor het programmeren van software niet meer per applicatie gaan, maar allemaal in Graphics Connect zitten, verandert er niet veel voor het programmeren van nieuwe software. Binnen Graphics Connect zullen de codes van alle overige aansturingsfuncties moeten worden uitgebreid met de XPressionSDK.

8. Discussie

Grote delen van dit onderzoek zijn direct of indirect gebaseerd op de gebruikers en de makers van de softwares. Hierdoor is veel informatie direct aangeleverd. In combinatie met de informatie van de experts, zijn er ook bronnen gebruikt van grote en kleine bedrijven/sites. Als dit onderzoek herhaald zou worden, wordt er verwacht dat de resultaten niet veel zullen verschillen.

Het onderzoeken van de functionaliteiten van Graphics Connect was lastig dan van tevoren werd ingeschat. Hierdoor had dit onderzoek met meer tijd een completer product kunnen opleveren. Op dit moment is er nog maar één functie van XPression geïmplementeerd in Graphics Connect. Het doel aan het begin van dit onderzoek was dat er een volledig werkende uitbreiding kwam op Graphics Connect. Dit is helaas niet gelukt, maar de functie die wel geïmplementeerd is, is tot een succesvol einde gebracht.

Een beperking van dit onderzoek was dat de libary die gemaakt is door Videohouse niet aangepast kon worden. Dit zorgde ervoor dat de 'Clear' functie die in Graphics Connect gemaakt werd voor de XPression uitbreiding, nu bij dezelfde switch case als van CasparCG in kwam. Het gevolg hiervan was dat als een gebruiker Graphics Connect start, deze niet meer kon kiezen voor CasparCG of XPression, maar beide tegelijkertijd. Tevens wordt Graphics Connect niet gebruikt door andere bedrijven, wat de verkrijgbaarheid van de informatie bemoeilijkt.

CasparCG en XPression zijn vanuit dezelfde switch case geschreven. Hierdoor voeren beide engines tegelijkertijd een functie uit. Het advies voor een vervolgonderzoek of uitbreiding op dit onderzoek is om te kijken of dit niet een betere optie is. Het voordeel hiervan zou zijn dat een eindgebruiker niet meer hoeft te wisselen tussen de verschillende render-engines tijdens producties en het wordt dan mogelijk om met een instantie van Graphics Connect tegelijkertijd meerdere engines aan te sturen.

9. Bronnen

- Adobe. (z.d.). Software voor visuele effecten en bewegende beelden | Adobe After Effects. Adobe After Effects. https://www.adobe.com/nl/products/aftereffects.html
- Avid. (z.d.). Broadcast Graphics Solutions [Illustratie]. Broadcast Graphics Solutions | Avid. https://www.avid.com/solutions/broadcast-graphics
  Bonestroo, W.J., Meesters, M., Niels, R., Schagen, J.D., Henneke, L., Turnhout, K. van (2018): ICT Research Methods. HBO-i, Amsterdam. ISBN/EAN: 9990002067426. Available from: http://www.ictresearchmethods.nl/
- CasparCG. (z.d.). GitHub. Geraadpleegd op 28 november 2020, van https://github.com/CasparCG
- CristiFati. (2015, 29 mei). how to edit or see the source code for dll files [Stackoverflow forumpost]. Stack Overflow. https://stackoverflow.com/questions/30534554/how-to-edit-or-see-the-source-code-for-dll-files
- Dynamic-link library. (2016, juni 26). Wikipedia, de vrije encyclopedie. Opgehaald 10:14, juni 26, 2016 van https://nl.wikipedia.org/w/index.php?title=Dynamic-link_library&oldid=46986146.
- Euro Media Group. (z.d.). Euro Media Group. Geraadpleegd op 5 december 2020, van https://www.euromediagroup.com/home
- HAN - University of Applied Sciences & AUAS - Amsterdam University of Applied Sciences. (z.d.). CMD Methods Pack - find a combination of research methods that suit your needs. CMD Methods Pack. Geraadpleegd op 9 januari 2021, van https://www.cmdmethods.nl/
- Lingo. (2021, januari 6). Wikipedia, de vrije encyclopedie. Opgehaald 18:41, januari 6, 2021 van https://nl.wikipedia.org/w/index.php?title=Lingo&oldid=57964753
- Microsoft. (z.d.). Socket.Connect Method (System.Net.Sockets). Microsoft Docs. Geraadpleegd op 10 januari 2021, van https://docs.microsoft.com/en-us/dotnet/api/system.net.sockets.socket.connect?view=net-5.0
- Microsoft. (2019, 9 augustus). Windows Forms Designer tutorial - Visual Studio. Microsoft Docs. https://docs.microsoft.com/en-us/visualstudio/designers/walkthrough-windows-forms-designer?view=vs-2019
- Microsoft. (z.d.). TcpClient.Connect Method (System.Net.Sockets). Microsoft Docs. Geraadpleegd op 10 januari 2021, van https://docs.microsoft.com/en-us/dotnet/api/system.net.sockets.tcpclient.connect?view=net-5.0
- Npostart. (2011, 31 oktober). Lingo [Video]. www.npostart.nl. https://www.npostart.nl/lingo/31-10-2011/TROS_1257191
- Ross Video. (2017, 29 maart). XPression U: Stagger Animations in XPression (Basics 134) [Video]. YouTube. https://www.youtube.com/watch?v=jLkbrFyn0Ps
- Valdellon, L. (2020, 30 juli). What is an SDK? Everything You Need to Know. CleverTap. https://clevertap.com/blog/what-is-an-sdk/
- Videohouse. (2020, 13 februari). Build software better, together (6.0.0) [Graphics Connect]. Videohouse. https://github.com/Videohouse/GraphicConnect/tree/master
- Visual Basic. (2020, december 2). Wikipedia, de vrije encyclopedie. Opgehaald 10:29, december 2, 2020 van https://nl.wikipedia.org/w/index.php?title=Visual_Basic&oldid=57662298.
- Vizrt. (z.d.). Vizrt. Geraadpleegd op 5 december 2020, van https://www.vizrt.com/
- Walker, D. (2001). The Evolution of Broadcast Graphics. Evolution and Trends of Digital Media, 1–7. https://evolutionofbroadcastgraphics.files.wordpress.com/2011/06/evolution-of-broadcast-graphics.pdf

- Wikipedia contributors. (2020, December 30). Enumerated type. In Wikipedia, The Free Encyclopedia. Retrieved 19:37, January 10, 2021, from https://en.wikipedia.org/w/index.php?title=Enumerated_type&oldid=997238094
- Wikipedia contributors. (2020, November 28). Dynamic-link library. In Wikipedia, The Free Encyclopedia. Retrieved 15:31, January 10, 2021, from https://en.wikipedia.org/w/index.php?title=Dynamic-link_library&oldid=991063906
- Wikipedia contributors. (2020, October 18). Key frame. In Wikipedia, The Free Encyclopedia. Retrieved 13:37, January 10, 2021, from https://en.wikipedia.org/w/index.php?title=Key_frame&oldid=984185845
- XPression | CG & Graphics Systems. (z.d.). Ross Video. Geraadpleegd op 28 november 2020, van https://www.rossvideo.com/products-services/acquisition-production/cg-graphics-systems/xpression/

10. Bijlagen

10.1 Bijlage A – Pdf van examples van XPressionSDK

Loading a project

Send comments on this topic.

XPression API Examples > Loading a project

Introduction

This example will show you how to create the objects to successfully load a project and display a scene using the BlueBox objects.

Example

[Visual Basic]

```
'create a new XPression engine instance dim
 Engine as new xpEngine
'dim a scene
 object dim Scene
 as xpScene

    'load a project file from disk   if
Engine.LoadProject("c:\demo.xpf") then

        'retrieve a scene object from the project we just loaded     if
Engine.GetSceneByName("scene1", Scene) then
            'set the scene online on framebuffer 0, layer 0
      Scene.SetOnline 0,
0     end if   end if
```


Using transitions

Send comments on this topic.

XPression API Examples > Using transitions

Introduction

This example will show you how use a transition object to transition a scene onto a framebuffer.

Example

[Visual Basic]

```
'this example assumes the existance of a formerly created engine with a loaded project dim Scene as xpScene
dim Factory as xpTransitionFactory 'the transition factory dim Transition
 as xpTransition 'the transition

    'retrieve the scene which we would like to transition online
  Engine.GetSceneByName "scene1", Scene

    'retrieve a transition factory
  Engine.GetTransitionFactory 0, Factory

    'retrieve a transition instance from the factory
  Factory.GetTransition 0, Transition

    'set the duration of the transition to 15 frames
  Transition.Duration = 15

    'set the scene online on framebuffer 0, using the transition   Scene.SetOnline 0, 0, -1,
Transition
```


Using framebuffer layers

Send comments on this topic.

XPression API Examples > Using framebuffer layers

Introduction

This example will show you how to display multiple scenes on a single framebuffer using layers.

Example

[Visual Basic]

```
'this example assumes the existance of a formerly created engine with a loaded project dim SceneBackground
 as xpScene 'the background scene dim SceneForeground as xpScene 'the foreground scene

    'retrieve the scenes
  Engine.GetSceneByName "scene1", SceneBackground
  Engine.GetSceneByName "scene2", SceneForeground

    'to prevent the background scene from being shown without the foreground scene,
    'we lock the render pipeline before we start setting the scenes online
  Engine.Lock

        'set the background scene online on framebuffer 0
    SceneBackground.SetOnline 0, -1 'use layer -1

        'set the foreground scene online on framebuffer 0
```

```
SceneForeground.SetOnline 0, 0 'use layer 0

'unlock the render pipeline so the two scenes will actually be rendered to the framebuffer
Engine.UnLock
```

NOTE: In the example above, the locking of the engine could be omitted. The locking is included since you would usually want the first frame rendered to be the composite of both scenes, the locking simply prevents output updates until we are ready sending BlueBox commands.

Displaying and starting a crawl

Send comments on this topic.

XPression API Examples > Displaying and starting a crawl

Introduction

This example will show you how to retrieve a crawl scene, set it online and start it.

Example

[Visual Basic]

```
'this example assumes the existance of a formerly created engine with a loaded project dim SceneCrawl as
xpSceneGroup 'note the use of xpSceneGroup

'retrieve the crawl scene from the project
Engine.GetSceneByName "democrawl", SceneCrawl

'enable "soft" starting of the crawl
SceneCrawl.SoftStart = True
SceneCrawl.SoftStartFrames = 20

'set the crawl scene online on framebuffer 0
SceneCrawl.SetOnline 0

'now start the crawl
SceneCrawl.Start
```

NOTE: The example above would be exactly the same for displaying and starting a roll instead of a crawl since a roll is also a xpSceneGroup object.

Accessing and traversing metadata

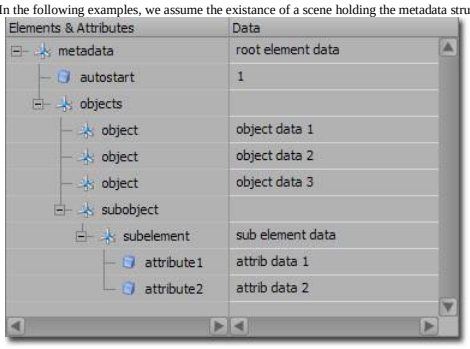
Send comments on this topic.

XPression API Examples > Accessing and traversing metadata

Introduction

This example will show you how retrieve and traverse the user defined Metadata using the COM objects.

In the following examples, we assume the existance of a scene holding the metadata structure shown in the image below:



Example

[Visual Basic] dim

```
mData as

xpMetadata

'retrieve the metadata object from the scene
'the current element will be set to the root of the internal metadata XML document
Scene.GetMetaData mData

'output the root element's name and data content to the debug window
Debug.Print mData.CurrentElement.Name 'the name: "metadata"
Debug.Print mData.CurrentElement.AsString 'the data: "root element data"

'show a messagebox when the current element's "autostart" attribute is TRUE
'in our example metadata the autostart value is "1" so the result will return TRUE  if
mData.GetAttribByName("autostart").AsBool = TRUE then
  MsgBox "autostart is true"  end
if

'set the current element to the "objects" node below the current element  mData.MoveToElement "objects", False

'traverse the "object" nodes below the current "objects" element
```

```
    'this will output the 3 "object" elements their name data values to the debug window   while
mData.NextElement("object")
    Debug.Print mData.CurrentElement.Name + " " + mData.CurrentElement.AsString   wend

    'for the sake of this example we now move back to the root of the metadata
    'and we directly navigate to the "subelement" node of the internal XML document   mData.SetRoot
  mData.MoveToElement "subelement", True 'move to the "subelement" node, recusively traversing all child elements

    'output the current element's attribute values   dim i as long   for i = 0 to mData.AttribCount - 1 'AttribCount will return 2 since our example data has 2 attributes defined for the
"subelement" node
    Debug.Print mData.GetAttrib(i).AsString   next
```


Using the APITexture Shader

Send comments on this topic.

XPression API Examples > Using the APITexture Shader

Introduction

These examples will show you how to use the APITexture shader to draw lines and primitives.

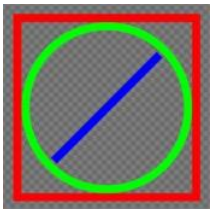
In the examples below it is assumed that a material named "Material1" exists in the project and this material should contain an APITexture shader named "ApiTexture". Please note that the xpPen, xpBrush and xpPointList object types are created (ie: CreatePen) as object instances

Example 1, Drawing simple primitives

[Visual Basic]

```
'create a new XPression engine instance
 dim Engine as new xpEngine 'dim a
 material variable dim Mat as xpMaterial
'dim an APITexture shader variable
 dim APITexture as
 xpAPITextureShader
'dim a Pen object variable
 dim Pen as xpPen

    'retrieve "material1" from the project (see assumptions above)
  Engine.GetMaterialByName("Material1", Mat)

  'retrieve the APITexture shader from the material
  Mat.GetShaderByName("APITexture", APITexture)

  'create the Pen instance
  Pen = Engine.CreatePen
  'set the Pen's size to 20
  Pen.Size = 20
  'set the Pen's color to blue and alpha to full
  Pen.SetColor(0, 0, 255, 255)

  'now draw a single blue line from xy coordinate 100,100 to xy coordinate 360,360
  APITexture.DrawLine(Pen, 100, 100, 360, 360)

  'set the Pen's color to green and alpha to full
     Pen.SetColor(0, 255, 0, 255)
 'draw a green circle with a diameter of 400 pixels at xy location 30,30
  APITexture.DrawEllipse(Pen, 30, 30, 400, 400)

  'set the Pen's color to red and alpha to full
     Pen.SetColor(255, 0, 0, 255)
 'and draw a red rectangle around the previous two primitives
  APITexture.DrawRect(Pen, 10, 10, 440, 440)   'the end result of the above code should now
```

be equal to the image below:

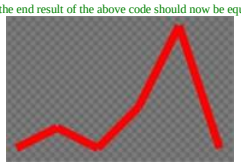


Example 2, Drawing Connected lines

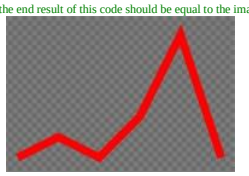
[Visual Basic]

```
'create a new XPression engine instance
 dim Engine as new xpEngine 'dim a
 material variable dim Mat as xpMaterial
'dim an APITexture shader variable
 dim APITexture as
 xpAPITextureShader
'dim a Pen object variable
 dim Pen as xpPen
'dim a PointList object variable dim
 Points as xpPointList

    'retrieve "material1" from the project (see assumptions above)
  Engine.GetMaterialByName("Material1", Mat)

  'retrieve the APITexture shader from the material
  Mat.GetShaderByName("APITexture", APITexture)
  'create the Pen instance
  Pen = Engine.CreatePen
  'set the Pen's size to 20
  Pen.Size = 20
  'set the Pen's color to red and alpha to full
     Pen.SetColor(255, 0, 0, 255)

  'now using (disconnected) line mode we draw a number of lines representing a graph
  'note: to prevent redrawing on each line instruction we call Begin/EndUpdate to improve performance    APITexture.BeginUpdate()
```

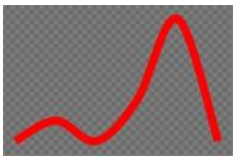
```
APITexture.MoveTo(100, 100)
APITexture.LineTo(Pen, 200, 150)
APITexture.LineTo(Pen, 300, 100)
APITexture.LineTo(Pen, 400, 200)
APITexture.LineTo(Pen, 500, 400)
APITexture.LineTo(Pen, 600, 100)
APITexture.EndUpdate()

'the end result of the above code should now be equal to the image below (note the disconnect between the lines):
```



```
'create the PointList instance
Points = Engine.CreatePointList
'to improve the above line connections one can use an xpPointList object and draw the lines using the DrawLines method like so:  Points.AddPoint(100, 100)
Points.AddPoint(200, 150)
Points.AddPoint(300, 100)
Points.AddPoint(400, 200)
Points.AddPoint(500, 400)
Points.AddPoint(600, 100)
'now draw the lines using the PointList
APITexture.DrawLines(Pen, Points)

'the end result of this code should be equal to the image below (note the closed connection between the lines):
```



```
'should you want to use curves instead of straight connected lines, you can simply reuse the PointList from above and call DrawCurve() instead:
APITexture.DrawCurves(Pen, Points)
  'the end result of using this curve code should be equal to the image below:
```



Reading keyframes from an Animation Controller

Send comments on this topic.

XPression API Examples > Reading keyframes from an Animation Controller

Introduction

To read keyframes back from an animation controller, you must first discover which objects have keyframes within that animation controller. The xpAnimController class contains a property ObjectCount and a method GetObject() to retrieve the objects from the animation controller.

With the objects in hand, you can retrieve a list of keyframe types that each object in the animation controller contains. The keyframe types represent a single property for example "Position.X", "Position.Y", or "Rotation.X". Not all object types will have the same keyframe types. For example, a Quad Object will not have a "Scale.Z" keyframe type. Given an object and a Keyframe Type you can proceed to call xpAnimController.GetKeyframe(). The GetKeyframe method will return a single keyframe object that represents the value for a single Keyframe Type at a single point in time. The Time value returned in the Keyframe object will tell you at what time the keyframe is located in the animation controller. This may not be exactly equal to the time you requested (but it will never be earlier in time than what you asked for.) You may proceed to call GetKeyFrame() multiple times to retrieve all of the keyframes for an object. Each time you call GetKeyFrame() you should set the Time value to one greater than the time of the previously retrieved keyframe.

Alternatively you can use GetFirstKeyframe() or GetLastKeyframe() to retrieve either the first or last keyframe for a given object in the animation controller.

Retrieving Keyframes from an Animation Controller — Copy Code

```
// This example will retrieve all keyframes for all objects and add them to a richTextBox for display private void button1_Click(object sender,
EventArgs e)
{
  xpAnimController animcontroller;
 xpScene scene;   xpBaseObject obj;
 xpKeyFrame keyframe;   int Time;
 richTextBox1.Text = "";

  // Get the scene and animation controller   engine.GetSceneByName("Scene1",
 out scene);   scene.GetAnimController(0, out animcontroller);

  // Loop over all keyframed objects
  for (int i = 0; i < animcontroller.ObjectCount; i++)
  {
    if (animcontroller.GetObject(i, out obj))
    {
      richTextBox1.AppendText("OBJECT: " + obj.Name + "\n");
```

```
    int links = animcontroller.GetKeyFrameTypeCount(obj);      for (int j = 0; j <
links; j++)
    {
     Time = 0;

      string LinkName;
      animcontroller.GetKeyFrameType(obj, j, out LinkName);
richTextBox1.AppendText(" " + LinkName + ":\n");
      while (animcontroller.GetKeyFrame(obj, LinkName, Time, out keyframe))       {
        // Add the Keyframes value and time to the text box
        richTextBox1.AppendText(" Time:" + keyframe.Time.ToString() + " Value = " + keyframe.Value.ToString() + "\n");
        Time = keyframe.Time + 1; // Start at the next position in time to get the next keyframe

}
}
    richTextBox1.AppendText("\n");
   }
  }
}
```


10.2 Bijlage B – C# Script van Videohouse met aansturing van Graphics Connect

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Diagnostics;
using Videohouse.Clients.GraphicConnect;
using Videohouse.Clients.CasparCG;
using Videohouse.Logging;
using Videohouse.GraphicConnect.Settings;
using Videohouse.GraphicConnect.Attributes;
using Videohouse.GraphicConnect.Commands.Console;
using Org.BouncyCastle.Utilities.Net;
using System.Net;
using System.Linq;

namespace Videohouse.GraphicConnect.Engines
{
    [GraphicConnectEngine]
    public class CasparCG : IExecutorOutput, IConsoleInfo
    {
        //TODO: Implement Responses
        #region FIELDS
        private readonly ICasparCGClient client;
        private readonly ILoggerManager loggerManager;
        #endregion

        #region CLASS PROPERTIES
        /// <summary>
        /// Gets the console information.
        /// </summary>
        /// <value>
        /// The console information.
        /// </value>
        string IConsoleInfo.ConsoleInfo => "Not defined";
        #endregion

        #region CLASS METHODS
        /// <summary>
        /// Initializes a new instance of the <see cref="CasparCG"/> class.
        /// </summary>
        /// <param name="client">The client.</param>
        /// <param name="loggerManager">The logger manager.</param>
        public CasparCG(ICasparCGClient client, ILoggerManager loggerManager, ISettingsManager settingsManager)
        {
            this.client = client;
            this.loggerManager = loggerManager;

            client.Connect(System.Net.IPAddress.Loopback.ToString(), 5250);

            loggerManager.Log("New Instance of CasparCG executor output", LogLevel.debug);
        }

        /// <summary>
        /// Cleans up the renders.
        /// </summary>
```

```
        /// <param name="Response">The response.</param>
        public void CleanUp(out Response Response)
        {
            List<Channel> channels = client.Channels;
            foreach(Channel channel in channels)
            {
                client.CGClear(channel.Index, out CasparCGCommand command);
            }
            Response = null;
            loggerManager.Log("Clean up the renders");
        }

        /// <summary>
        /// Clears the specified channel.
        /// </summary>
        /// <param name="channel">The channel.</param>
        /// <param name="layer">The layer.</param>
        /// <param name="response">The response.</param>
        public void Clear(int channel, int layer, out Response response)
        {
            client.CGClear(channel + 1, layer + 1, out CasparCGCommand command);
            response = null;
            loggerManager.Log(string.Format("Clear channel {0} and layer {1}", channel, layer), LogLevel.info);
        }

        /// <summary>
        /// Updates the currently loaded template at the specified channel very fast. Use this method if the engine
has an option to send lots of data.
        /// </summary>
        /// <param name="channel">The channel.</param>
        /// <param name="key">The key.</param>
        /// <param name="value">The value.</param>
        public void FastUpdate(int channel, string key, string value)
        {
            IEnumerable<IDataCollectionItem> dataCollection = new List<IDataCollectionItem>() { new
DataCollectionItem(key, value) };
            Update(channel, 0, dataCollection, out Response response);
        }

        /// <summary>
        /// Updates the currently loaded template at every channel very fast. Use this method if the engine has an
option to send lots of data.
        /// </summary>
        /// <param name="Key">The key.</param>
        /// <param name="Value">The value.</param>
        public void FastUpdate(string Key, string Value)
        {
            this.FastUpdate(1, Key, Value);
        }

        /// <summary>
        /// Initializes this instance.
        /// </summary>
        /// <param name="Response">The response.</param>
        /// <exception cref="NotImplementedException"></exception>
        public void Initialize(out Response response)
        {
            response = null;
            loggerManager.Log("Initialize not implemented", LogLevel.info);
        }

        /// <summary>
        /// Loads the specified template at the specified channel and layer.
        /// </summary>
        /// <param name="channel">The channel.</param>
        /// <param name="layer">The layer.</param>
        /// <param name="template">The template.</param>
        /// <param name="playOnLoad">if set to <c>true</c> [play on load].</param>
        /// <param name="dataCollection">The data collection.</param>
        /// <param name="response">The response.</param>
        public void Load(int channel, int layer, string template, bool playOnLoad, IEnumerable<IDataCollectionItem>
dataCollection, out Response response)
        {
            response = null;
            loggerManager.Log(string.Format("Load command channel {0}, layer {1}, template {2}", channel, layer,
template), LogLevel.info);
            client.CGAdd(channel + 1, layer + 1, 1, template, playOnLoad, dataCollection.ToList(), out
CasparCGCommand command);
        }

        /// <summary>
        /// Resume animation of the currently loaded template at the specified channel and layer.
        /// </summary>
        /// <param name="channel">The channel.</param>
        /// <param name="layer">The layer.</param>
```

```
        /// <param name="response">The response.</param>
        public void Next(int channel, int layer, out Response response)
        {
            client.CGNext(channel + 1, layer + 1, 1, out CasparCGCommand command);
            response = null;
            loggerManager.Log(string.Format("Next command channel {0} and layer {1}", channel, layer),
LogLevel.info);
        }

        /// <summary>
        /// Plays the currently loaded template at the specified channel and layer.
        /// </summary>
        /// <param name="channel">The channel.</param>
        /// <param name="layer">The layer.</param>
        /// <param name="response">The response.</param>
        public void Play(int channel, int layer, out Response response)
        {
            client.CGPlay(channel + 1, layer + 1, 1, out CasparCGCommand command);
            response = null;
            loggerManager.Log(string.Format("Play command channel {0} and layer {1}", channel, layer),
LogLevel.info);
        }

        /// <summary>
        /// Sets the path.
        /// </summary>
        /// <param name="path">The path.</param>
        /// <param name="response">The response.</param>
        public void SetPath(string path, out Response response)
        {
            client.Path = path;
            response = null;
            loggerManager.Log("SetPath: " + path, LogLevel.info);
        }

        /// <summary>
        /// Enable test outputs on the engine.
        /// </summary>
        /// <param name="Text">The text.</param>
        /// <param name="Response">The response.</param>
        public void TestOutputs(string text, out Response response)
        {
            FileVersionInfo info = FileVersionInfo.GetVersionInfo(GetType().Assembly.Location);
            string companyName = info.CompanyName;
            foreach(Channel channel in client.Channels)
            {
                IList<IDataCollectionItem> dataCollection = new List<IDataCollectionItem>
                {
                    new DataCollectionItem("Name", client.Name),
                    new DataCollectionItem("Channel", channel.Index),
                    new DataCollectionItem("Machine", Environment.MachineName),
                    new DataCollectionItem("Company", companyName),
                    new DataCollectionItem("VideoMode", channel.VideoMode),
                    new DataCollectionItem("Custom", text)
                };

                //clear channel
                client.CGClear(channel.Index, out CasparCGCommand command);

                //play template on channel
                client.CGAdd(channel.Index, 1, 1, "colorbars", true, dataCollection, out  command);
            }
            response = null;
        }

        /// <summary>
        /// Updates the currently loaded template at the specified channel and layer.
        /// </summary>
        /// <param name="channel">The channel.</param>
        /// <param name="layer">The layer.</param>
        /// <param name="dataCollection">The data collection.</param>
        /// <param name="Response">The response.</param>
        public void Update(int channel, int layer, IEnumerable<IDataCollectionItem> dataCollection, out Response
Response)
        {
            client.CGUpdate(channel + 1, layer + 1, 1, dataCollection, out CasparCGCommand command);
            Response = null;
            loggerManager.Log(string.Format("Update command channel {0} and layer {1}", channel, layer),
LogLevel.info);
        }

        /// <summary>
        /// Updates the camera
        /// </summary>
        /// <param name="Channel">The channel.</param>
```

```
        /// <param name="Camera">The camera.</param>
        /// <param name="Response">The response.</param>
        public void UpdateCamera(int Channel, Camera Camera, out Response Response)
        {
            Response = null;
            loggerManager.Log("UpdateCamera not implemented", LogLevel.error);
        }
        #endregion
    }
}
```