Technology Review

February 12, 2017

Abstract

The Forge VR Explorer branches from an Autodesk prototype project called Vrok-It, which is a simple web-based 3D model viewer and mobile virtual reality (VR) explorer. The project will expand upon its ability to display uploaded 3D models in browser and in VR, and improve its accessibility. Conventionally, viewing 3D models in VR is a challenge if you have model files on many devices, or have a headset that only works in conjunction with a smart-phone. The Forge VR Explorer aims to do this by utilizing a web-based software that uses the features of the Autodesk Forge API. The project will also be expanded with new ideas and stretch goals as the project is developed.

TABLE OF CONTENTS

1 INTRODUCTION

The Forge VR team (group 20) will be working building on the Vrok-It project. We will be trying to make the site more accessible for users as well as trying to increase the performance of the of the viewer and the VR experience too. Shawn Cross will be tasked with the file uploading feature, file conversion, and connecting the users device. Griffin Gonsalves will be working on the Website UI Redesign, Data Management API and Model Viewer. Paul Kwak is going to be working on Forge Authentication, Viewing in VR, and hardware detection.

2 TECHNOLOGIES

2.1 File Uploading Locally

1) Options

- Use a plug-in: Using a plug-in that would handle file uploading would be very convenient as it takes a lot of the work out of it. One benefit of using a plug-in is that this would save us a lot of time. If we were to use the plug-in we would not have to worry about the effort that would normally go into creating our own, we could simply just find one that would fit what we needed and the put it into our website. However this does also lead to the problem that we may not be able to find one that exactly fits what we want. Many of the plug-ins I have found so far simply upload the file to a server and then it is done. In our case we need the file to be usable by the website once it is uploaded.
- Create our own file upload feature: Creating our own uploading feature would allow us to make the feature work exactly the way that we would want to. Their a many different ways to do this though. One common way I have found is to first create a web form using HTML to make the upload button.Then you could use PHP to submit the file through a POST. Using AJAX and JQuery you could also add features that show the progress of the upload and verification such as verifying correct file types are being used. This could take a lot more time and effort to complete though depending on how skilled the person creating the upload feature is and how fancy they want the feature to be.
- Create our own file uploading feature using new File API added to HTML5: This is similar to the previous option but seems like it might be easier to learn and understand. Many of the features that we wanted are built into the API making it easier for us to create the upload file file feature the way we want it. Though there will be some time for learning how to use the API and some time to completely create the feature, there is a lot of good documentation on this API that should make the process go smoothly.

2) Criteria being evaluated

   a) How much time will it take to integrate into our project: The plug-in will definitely take the least amount of time to add this to the project. I think that the other two options will both take more time to do since you will have to create the upload feature from scratch.
   b) How much effort will have to go into learning how to complete the task: The plug-in won't take a lot of time to learn how to use since you just need to add the library to the project and then you can use it. The other two options will take more time to learn how to do but shouldn't be to much time since there is a lot of documentation and examples on-line showing how to do this.
   c) Will the upload feature do exactly what you wanted it to do: If you aren't really particular about what the file upload button looks like or how it shows the progress of the upload then the plug-in would work just fine. However if you want to make the upload button look a certain way and show its progress a certain way then one of the other two options is going to be best.
   d) Which option will offer the most secure way to upload the file: All three options offer the same amount of security.

3) Discussion The plug-in option will definitely save the most time in terms of learning it and getting it to work in on the website, however it will likely give you little in the way of customizing and making the file upload feature work exactly how you would want it to. Both of the options to make you own file upload feature will result in a longer time to learn how to do this as well as a longer time to complete the task. This does allow you to make sure the the file upload will do exactly what you are wanting it to do. the third option will is also using HTML5 this means that is will work in all the new browsers.
4) Select Option The custom file api is the best option to go with. While this option may take a little extra time the flexibility of being a able to make it exactly the way we want allows us to be more restrictive on what file will be allowed to be uploaded. This option also allows up to make this part of the website look exactly how we want it.

2.2 File Uploading Non-Local

1) Options

   - The first option would be to create this from scratch. First this would require that we have a good knowledge and understanding of many different development tools such as Javascript, JSON, Node.js, and possibly others. This would also require a lot of code to be written and that would require a significant amount of time. This option would allow us to make this file system exactly the way we want it. Since the Forge apis also do not have a GUI for going through the users files. We would need to research and learn how to implement something like this and that would take time.

- The other option is integrate an api that has already been made into the system we have in place. The api that I have found is called model.derivative-nodejs-sample [**?**] and was created by Adam Nagy. The api we has a file interface that looks a lot like what we were wanting it to. We would need a good understanding of the technologies used in the api since we will have to integrate into the system we currently have. The cost of time of this option is greatly reduced compared to the previous option. We also have found that Kean would be able to get a hold of the person that has built the api to help us to get the integration done.

2) Criteria being evaluated

   a) Time it will take to implement this task. The first project will take time to research and learn any of the technologies that we don't know or understand. It will take a large amount of time to actually code this complete task. Creating the interface that the user would use to navigate to the files they are looking for would also take time to learn how and to implement. The second option would take a small to medium amount of time to integrate the api into our current system. This will require us to learn the different technologies that are being used in the api however with the help of the person that wrote the api could make that learning curve a lot easier taking only a small amount of time.

   b) The option to make our own would allow us to make the file upload system exactly the way we want this. This makes it so the we make the interface that users will use to get to their files exactly the way that we want it to. The second option doesn't allow for as much customization. We could probably figure out how to change the appearance but this would cost more time trying to learn how to do that.

3) Discussion The first option would allow us a lot more freedom to make the file interface the way that we want it to however this will require not only a lot of research of to learn how to get all of the forge apis working but it would also require us to learn how to create the file interface system itself. The api that we have found has a look that all of us in the group agree that it looks like what we were wanting to do. The time factor for making our own will be very significantly greater than if we chose to go with the second option.

4) Select Option We will be using the second option since we only have a fairly limited amount of time to get this project finished. Even though it may not be exactly what we want the interface to look like it is an interface that is easy for someone to understand how to navigate which is what we most want.

2.3 File conversion

1) Options

- accusoft API: [**?**] This is an API that allows users to convert their CAD files in to raster files such as SVG (scalable vector graphic) files that could to view in the model viewer. This API does cost money and depending on how many conversions the user is planning on doing this option could turn out to be relatively expensive. There is a lot of documentation on how this API can be integrated into an application but it seems as though the integration process could be harder if you are not using a certain types of applications.
- Forge Model derivative API: [**?**] This API has the ability to take users CAD source file and convert them into OBJ and STL files. This API can also convert the files directly into SVF files the same type of file that the model viewer API we will be using on this project uses. This API would also not cost us anything to use while we are working on the project since the company that created the API is the same company we will be making the project for. There might be a little bit of a learning curve when it comes to integrating it into the website but the API has a lot of tutorials and documentation on how to use the API in an application. This API also uses a token based authentication system that provides security when converting the users files.
- cloudconvert API: [**?**] This API gives you the ability to convert from CAD source files to a SVG file but it also give you the ability to convert the other way was well. This API also has well documented instruction on how to integrate this API into an application. That would make it easier to learn and take less time to complete the task. This API would cost money to use depending on how much time is spent doing the conversions. this one also requires that a key be used when doing the to be created which make this more secure to do file conversions on since it will be able to verify that it is an actual users trying to do conversions.

2) Criteria being evaluated

   a) How much time will it take to integrate into the project: All three of the projects will take about the same amount of time to integrate into the project. I think that the first option might take longer since the documentation looked like it might be hard to understand and didn't seem as accessible as the other two APIs.
   b) How much effort will have to go into learning how to complete the task: I think each of these could take a good amount of effort to understand how to use and set up. Options two and three seem to have easier to understand documentation and have more example of how to use the API.
   c) How secure is the file conversion: The first API didn't say whether or not that it had a way of making sure that the file conversions were secure. Options two and three both had some sort of way of authenticating the conversion. Option two used a token system and option three used a key system.

3) Discussion: These three options all offer similar conversions. They all convert the CAD source files into a usable file for the model viewer. The Forge API will however convert the file into an SVF file that can then be directly loaded into the viewer. This could save time and make it easier to complete the task. The Forge API is also free for us to use during the duration of the project so that would also make a big difference as well. Finally the Forge API offers the most secure option when it comes to converting the users files since it requires a token authentication to even do the conversion. This token is created as soon as the user visits the website and last for the duration that the user is on the website. Also with lots of documentation as well as tutorials on how to integrate the API into our application I feel like this option will take the least amount of time to complete the task.
4) Select Option: I Think that we should use the Forge model derivative API since it seems the most straight forward on how to integrate it into our application. It won't cost us any money to use and has a more secure way of converting the users files.

2.4 Connecting the users device

1) Options:
    - Use a QR code plug-in: jquery.qrcode.js is a jquery plug-in that generates a QR code which can be scanned by a user to using a QR scanning application on their device to connect their device to the current web session. Using this would be relatively easy since we would just need to add the library to our project. That means that there wouldn't be a lot of learning time involved in using this option.
    - QRickit API: This API can dynamically generate QR codes to be used in a web/mobile application. The API is free to use for personal and small business applications so there would be no cost for us to use it with our project. There is also documentation on how to setup and use the API so this would cut down on the time that it take to learn how to integrate it into our project.
    - QR Code Generator: This is another API that will that will generate a QR code for the us. It can easily be embedded into the web page by putting the source into an image tag. This API also allow for you to specify how big you want it and allows for different backgrounds to be used. The API also Has all the documentation that is needed to set up and use so the time to integrate this API into the project would be much.
2) Criteria being evaluated
    a) How much time will it take to integrate into the project: I think that all of the options can pretty easily be integrated into the project. all have documentation that clearly describes how set-up and use them. Overall I would say that each of them are about the same when it comes to the amount of time that it will take to integrate them into our project.

b) Will there be a lot of learning involved in using the option: I also feel like that since each of these options has documentation describing how to use each that the learning involved in each is pretty minimal.

c) Will it connect the users device quickly: each of the should have about the same time to connect the device. The only reason that there would be a slow down in connecting the users device is if they had a slow Internet connection or if there was a lot of traffic on the website itself.

3) Discussion: Overall I think that all the options are pretty close to one another. All three of the options are pretty straight forward in terms of integrating them into the project. All three should also connect the users device quickly. The two API options add the ability to customize the QR code, but that really isn't needed for our project since we just need the QR code to be there so the user can connect their device.

4) Select Option: I would say that the plug-in is the best way to go since we can integrate the library directly into our project and not have to worry about making any calls to an outside API.

2.5 Website UI Redesign

1) Options

- WordPress WordPress is a high-level framework that can generate a simple view for a website. Wordpress has the advantage of using many pre-created templates in order to display a page. Wordpress is highly supported by many browsers, and could simplify the process of creating updates to the site. There is also the added benefit of a very pristine looking webpage without spending too much time developing it.
- AngularJS Redesigning the entire site does not require starting from the ground up. We can implement a javascript framework that would help us organize and manage the site. Angular JS would assist us in creating a better interface for the website without that much learning required. Our project is already using javascript to handle much of the project operations, but implementing a framework for the interface should not conflict with the rest of the project. Angular is completely free and has plenty of documentation available. There is also the newer edition named Angular 2, but due to the complexity, it may not be advisable to use.
- ReactJS React JS is another javascript framework that can generate different views for the page from a special source file. It simplifies javascript programming, in addition to generating simplified code for the project. React is also completely free, and has plenty of documentation and tutorials available online. Mobile browsers also are supported by React.

2) Goals Improve site layout and visual appeal

3) Criteria being evaluated

- Development time: Each different option would potentially take different amounts of time to complete-potentially much longer than we have time for. Evaluating how long it will take to implement is a necessity.
- Presentation: This would be the look and style of the option. Specifically, whether it looks nice or not.
- Learning curve: Having to learn a new framework for web development could take weeks of time. This also factors into the development time.

4) Discussion Each option provides the advantage of creating a nice interface, but each does so at a cost. Angular could take the longest to learn, and could also be difficult to implement. React has the advantage of creating nicer interfaces at little overhead. It certainly is the most different and could be a challenge to learn, but the resulting pages could bet a great choice for the project. Wordpress, though convenient, will not be able to keep up with our project, and would be challenging to continually update the layout with each change to the project during the development phase.
5) Select Option In weighing the options, using ReactJS would be the best option for the project. Because the site already has a working page, we think it would benefit the project more to spend less time developing the updated UI and more time on the focus of our project. After learning the basics, ReactJS should provide a great interface for our website that improves our development process.

2.6 Data Management API

1) Option

- Forge Data Management API: Belonging to the Forge collection of APIs created by Autodesk, this software allows for the user to pull project files from an Autodesk A360 library that contains a user's project files. These files could be filtered and shown in a user interface, which is a criteria for this technology. Once a website is registered with the API, and a user is logged in, files may be accessed in the interface. The usage of the API is straightforward, after setup with the API, we would only need to include a provided configuration and setup on the site which could assist in rapidly developing this feature.
- OneDrive API: Another common cloud service is Microsoft's OneDrive. Microsoft provides a File Picker API through their Javascript SDK, which is also the main scripting language used in our project. After registering the app with Microsoft's API, one may implement a "Open from OneDrive" button which opens a File Picking window where the user may pick a file from their drive. This API would support all major web browsers, in addition to the most current mobile

browsers. This API should have no problem integrating with our other assets, and provides a UI popup for selecting the file.

- Google Drive API: A similar, but more general use case API that fits the bill is the Google Drive API. This API lets a user log into their google account and access their files stored on Google Drive. Google claims the API works flawlessly in a straightforward HTML web document, so compatibility should not be an issue with other components in the project. Furthermore, since the API supports the viewing and interaction with user's files in an interface, the API meets the criteria of having an interface. This interface, called the Google Picker, functions as the interface for picking the file.

2) Goals To provide an alternate method of file upload to the project through a cloud service.
3) Criteria being evaluated

- Level of browser/device support: How many modern browsers does the technology support?
- Project asset/Website compatibility: Does the new API work with the rest of the website?
- File picking user interface: Does the API's user-interface function for our project?

4) Discussion: This sounds like a cut and dry choice for the Forge API, as we are already using Autodesk APIs and libraries on the site in addition to other Forge components. Using this API could save development time as the app need only be registered with the Forge service once, and appears relatively trivial to implement in the site. The Microsoft API seems useful, but the OneDrive cloud service is not oriented to provide large files on demand, and configuration seems much more complex. On the other hand, the Google Drive API seems like it could be a suitable backup. The functionality seems slightly more complex, however this would provide cloud access for users on the most used cloud platform.
5) Select Option: Due to the inertia behind our project using the Forge Collection, I think the best option for the project is to include the Forge Data Management API. It meets each of the criteria for the project and should function as a useful feature when completed.

2.7 Model Viewer

1) Option

- Forge Large Model Viewer: This is the current system in place for the project. Created by Autodesk, the LMV allows for larger project files to be viewed and manipulated in-browser, and on mobile for most modern smartphones. The viewer is placed in the web document as a window.
- WebVR Rendering: A very new initiative from Mozilla that provides high performance graphics on the web aimed at supporting many new VR devices, including Google Cardboard. The system

supports different mobile devices and is also supported on mobile web browsers. The software is implemented within the project as a library.

- Unity 3D: Unity provides another method of 3D model viewing for browsers. The Unity Web Player is capable of detailed 3D projects in a web browser, including full 3D games, models, and more. Unity has also recently provided support for the Oculus Rift. However, the software is built upon the same OpenGL renderer as both WebVR and the LMV.

2) Goals Provide a viewing interface in which to view a 3D model in-browser.
3) Criteria being evaluated:

   - Model Quality:
   - Web Viewing Experience
   - VR Viewing Experience
   - Device and VR system support

4) Discussion: There is a large project codebase that was built on top of the Large Model Viewer specifically for the project. Moving to a different 3D/VR framework would take considerable work and commitment for the project as a whole, but could potentially bring more support and features to the project that we could use. WebVR by far seems the most advanced, high-quality and broad ranging in terms of device and browser support, but with the disadvantage of having nearly no documentation or tutorials. Unity provides a very easy to use web model viewer, but only has support for a single VR headset, which would fail to meet the Device and VR system support criteria.
5) Select Option: Currently the best option for our project at the moment would be the Large Model Viewer. Granted that both webVR and Unity support high quality models that can be inpsected closely, the amount of the project that currently relies on the LMV would be too large of a task to make the jump to a newer framework at this point given the alternatives. Through communication with our sponsor, we may be able to use a different framework later down the line, as there could be a new API to use. Currently, it does not seem sufficiently justified

2.8 Forge Authentication

1) Option

   - Write code from scratch. There are many ways to do this, specifically using XMLHTTPrequests or ajax to listen and send information. These methods would then be received on a node.js server. The client side code would utilize javascript to interact when buttons on the website are pressed.
   - Adapt current and existing implementations of the Forge OAuth2 API into the already written Vrok.it code. The implementation under consideration uses node.js like Vrok.it already for the

server hosting and also uses the express.js framework. It also contains working 3 legged authorization. The code also contains working Data Management and Model Derivative API which can be utilized on other sections of the project.

2) Goals Correctly redirect the user to the Autodesk site and receive tokens.
3) Criteria being evaluated
   - The amount of time necessary to write or rewrite the code and then test that code.
   - The ease of integration into Vrok.it's existing code.
   - Experience necessary to write code.
4) Discussion: The best things writing code from scratch provide is better understanding of the code, easier integration with already written Vrok.it and lower possiblity of code conflicts. Code written for the express purpose of working with Vrok.it, will do everything necessary with nothing extra unless needed. With code that has no extraneous lines leads to less errors due to unwanted interactions. The disadvantages to scratch writing code would be the lack of experience the group has with Javascript. Furthermore, with time constraints it becomes more difficult to write code from scratch as most, if not all of it will have to be researched and then written ground up.
   Adapting the already written API has one extremely severe drawback, it must be integrated into Vrok.it's existing code. It is entirely possible that the code may not even be compatible with Vrok.it. However, there are many advantages to adapting the existing code. The documentation is already there, the time necessary to adapt the code should be significantly less than writing from scratch and the accessible Data Management and Model Derivative API's help cut down on man hours.
5) Select Option : Adapt already written code. The time saving benefits of not having to write three sets of code for each API are immense. Furthermore, the code as currently written is testable, whereas the scratch code could take time before reaching a testable state.

2.9 Viewing in VR

1) Option
   - Google Cardboard: Costs $15. Turns any mobile device into a 3 dimensional viewing environment. Putting the phone into the contraption and using compatible apps allows a VR viewing experience. When porting in model files via QR code the issue is getting the files into format that makes them VR. There are a number of applications available for free that do this so the process might be very simple and definitely already exists.
   - Costs $599. A mobile VR viewer already exists that has CAD file viewing. However, costs money to use and is still a work in progress. Porting Solidworks CAD files directly into Unity is possible

and works but does not utilize the flow we are using. Another avenue with might be to use an application under the sketchfab VR umbrella.

- Costs $1,500. Requires a PC to either view on, but also requires the user's phone to have bluetooth if the user wishes to use the phone. With this option also changes the flow for the project. CAD files can be exported from Solidworks into a .3DS format. The .3DS format can be directly imported into Unity which then has Vive support and allows for viewing via that.

2) Goals display our adjusted model in VR while maintaining the smoothest possible viewing experience.
3) Criteria being evaluated

   - Cost/Accessibility
   - Documentation for each
   - How well does it fit within our established system.

4) Discussion: Google Cardboard has several advantages. First off it's incredibly cheap price, will allow for greater accessibility. Especially for developers, each us will be able to use our own and testing models should be easy. Second is the high number of application that display models and their easy access (free). This gives us an easy way to compare our work, framerate wise. Finally since our current setup utilizes Vrot.it and already had a QR scanner to port the models to our phones. It becomes a matter of adjusting the model and then using the VR format Cardboard uses to get our desired VR model. Oculus and Vive come with the advantage of having higher hardware to work with. Our need to detect hardware becomes less of an issue since both work with PC's and then display to the phone via other methods. This comes at their high cost and additionally the applications out for those options don't utilize our current setup. In fact would change our setup completely since they alleviate the need to even upload files and just change our problem to converting CAD files to work with Unity.
5) Select Option: Google Cardboard is significantly cheaper and perfectly fits within the way we have setup in order to utilize what we have.

3 CONCLUSION

After our technology review, we have made some important evaluations on our project and its different components. For many components, we decided it was best if we continued using some of the technologies from the original project, in addition to focusing on utilizing new API's and frameworks that fit the goals of our project. We are planning on potentially adding an updated version of the Forge Large Model Viewer that is currently being developed.