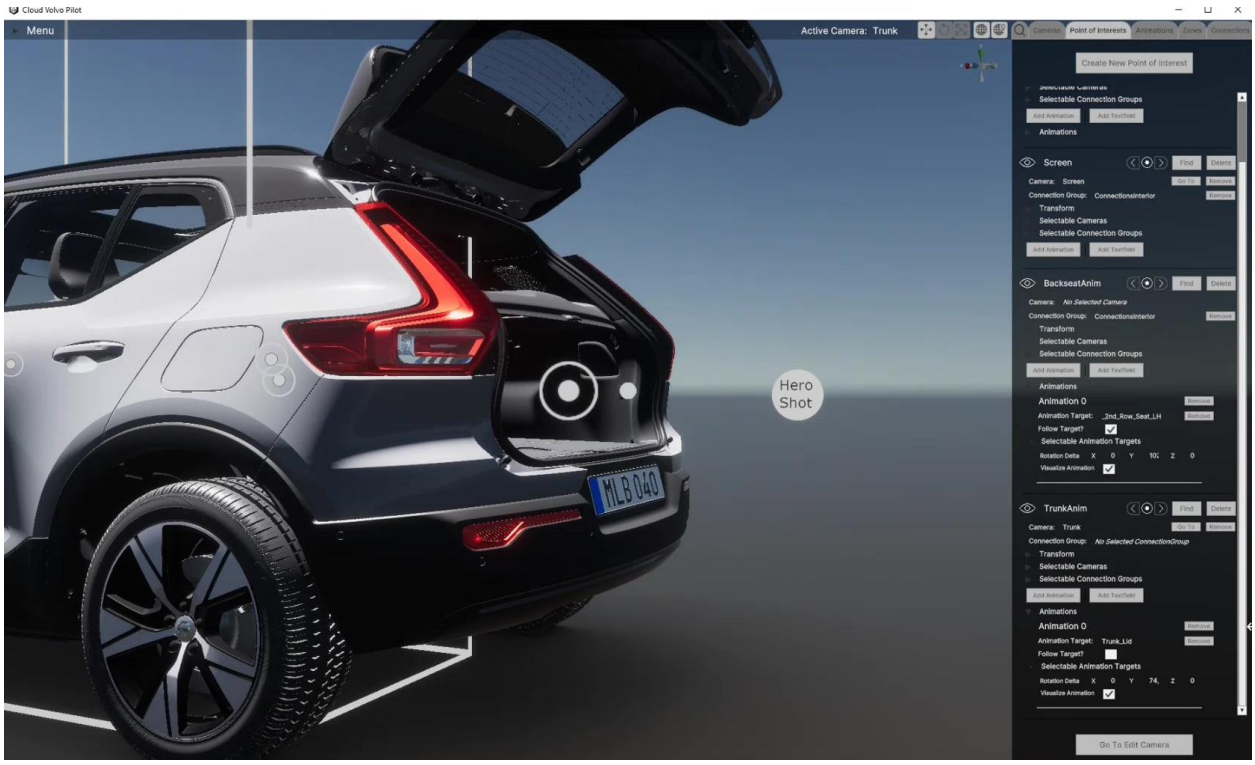
Developing a standalone application in Unity



YRGO, Industrial Technical Artist

Caroline Wallbäck

Gothenburg, Sweden

Spring 2022

Preface

I want to start by giving my thanks to Volvo Cars Gothenburg for giving me this opportunity to deepdive into coding in C#. Especially thanks to Kristoffer Helander, Technical Artist at Technical Visualisation Department at Volvo Cars, and my mentor, during this period, for helping through confusing times amongst all the ups and downs.

I also could not have done this without my classmate, and also a colleague at Volvo Cars, who also had the internship together with me, Sofia Friberg.

Table of Content

Introduction

Volvo Cars has an online-experience of viewing a car on their website. Any customer can overview a car in 3D-space, navigate in and around it, exploring it from specific angles, both exterior and interior. Currently this experience has only been made for one car model, involving lots of different people with different expertises. Developers for coding in Unity, and designers for deciding how the end result should look like. Now there has been a request for making this experience for several other car models, and Volvo is looking for a more time-efficient and easy way of designing these experiences, and without having to involve as many people from different departments.

Purpose

Our aim is to create an easy understandable and user friendly application, for in-house designers to use when designing the experience of viewing a car online on Volvo Cars website. The end goal for the user of this application is to create cameras in and around the car at specific spots and angles, and implement specific settings for what it will show and how it will behave. All this data needs to be exported into a Json-file which is used when the experience is implemented on the website. The purpose with the standalone application is that it can be used without any developing skills, so there won't be any need for developers in making the final experience setup.

Background

Originally this is made directly in Unity, placing all necessary objects in the scene and creating all functionality using code. This makes it complicated. The ones in design and marketing deciding what the user experience should look like don't have the right knowledge of Unity. But developers with the knowledge don't know what the experience should look like. The idea of making a separate software for this purpose has been up for a while at Volvo Cars but not yet done. A lot of code for functionality, like different camera-behaviours and settings, animations and more, is already done from making the original experience in Unity. We need to implement them in the application and make sure the user can create new objects, edit values and settings during runtime.

Questions and Limitations

‘How can we develop a new separate software with this already existing workflow from Unity, making it accessible and understandable for anyone at Volvo Cars to use?’

The first and biggest struggle for this project is that neither me, nor my team associate, had ever used Unity before and had very little coding experience, both in general but also specifically coding for Unity. Therefore our first question was of course ‘Will we be able to get something real and professionally done that is valuable in the end?’.

Then there have been a lot of questions and discussions, both from start but also throughout the project, around who will be the user for this software. We would want to get the perfect balance between rather complicated functionalities but also with an easy understandable user experience. The ultimate product would work differently depending on if the user knows the basics in Unity or similar softwares, and recognises for example navigation, functionalities and specific terms.

We had some doubts about how extensive this software can and/or should be. Which of all features do we need to include? The car would still need prepping in Unity with collision, materials, animations etc before it goes into this new application. What more will be needed to do in Unity and which features can we implement in the application?
While at it, can we make this software completely as a standalone application so Unity isn’t needed at all?

One problem we discovered when we started implementing more complicated functionalities caused limitations throughout the project. All already existing code for the original experience in Unity is designed and written with the requirement of placing everything in the scene and making all important connections in the Unity Editor before starting playmode. We needed to make sure that everything works even when you start the application with an empty scene and place all objects during runtime instead. This has been a challenge, since we had to rewrite existing and fairly important code and didn't always know what we could change in order for the existing functionality to still work, since it was someone else's code.

Implementation

Our first weeks at Volvo Cars we focused on exploring Unity, watching tutorials and reading documentation to get to know Unity and also coding in C# with Visual Studio Code. We studied the original project and read existing code to get an understanding of how it's built and works.

Before we started with the application, we did a simple user journey of the upcoming application, to get an idea of what it would look like and what to start with, and then presented it to our mentors to compare with their idea of this application to see if we had understood the assignment fairly correctly.

When starting with the project we began with a simple GUI with Unity UI Builder, since the whole project depends on the user interface. We implemented basic features such as moving around in the scene, creating cameras (placing objects in the scene containing the existing camera-scripts), deleting cameras, editing name and position with user input. We put a lot of focus on functionality for exporting the data to a Json-file since it's the end-goal when using this application. The Json-file needs to contain all relevant data to be able to get the experience up on the website. We also needed support for importing the data back to the application, so the user can save and import data to continue another day.

After a few weeks, we had a functioning application with the most basic features; placing the cameras at the desired position, displaying a list of all cameras in the scene, and exporting the data. Next step was to do a need analysis from the end-users perspective to be able to plan which features we needed and which were more prioritised. We did this together with two colleagues, one technical artist and one visualisation artist, to get a good balance between the requests from a designer and the more technical aspect of what is possible and not. From this, we put up a tasklog and planned a few weeks ahead at a time. We began with the most important features to get a complete application with all required components, then continued with the prioritised tasks that were not necessary for a working application but good-to-have-features.

We made the functionality of hovering over objects and clicking on them, and the functionality of editing the transform of the objects in the GUI, but we wanted a more intuitive way of editing the transform. What we wanted was classic transform handles which you have in pretty much any 3D programme. We tried coming up with different ways of doing them but we ended up downloading a finished version with 3D-models and scripts, since it was way easier and more time-efficient. We just adjusted some small stuff to make it work with our application. It was the same with the scene gizmo that shows the X-, Y- and Z-axis of the world.

Since this application will be used for several different car models the idea from start was to make a new build of the application for each car, with the car already in the scene. We managed to make a solution for importing any car directly in the application, with asset bundles in Unity, so there is no need to ever rebuild the application. We also made a solution for making the animations of car parts in the application so it wouldn't be needed in the preparation of the car model in Unity.

When we had a lot of implemented functionality, such as placing all necessary objects (cameras, points of interest, zones, connection groups) and displaying them in separate lists, moving around in the scene, editing all settings, switching between edit mode and preview mode, exporting and importing data, we did some user tests with our colleagues (in Technical and Design Visualisation) to get useful information about the GUI and if they could understand and use the application intuitively.

We evaluated all feedback and decided what was more prioritised to implement, then adjusted and added some functionality, for example we changed the GUI a bit to get it more intuitive to work with and added tooltips.

Before we handed over the final product we went through all the code. We made sure there wasn't any unused code and commented on the code so the next person that will pick up this project, or if something needs adjustments, easily will understand what the code does. We also made documentation of everything which contains how asset bundles work that we used for importing the car, how the application works, and information about all the settings..

Result

The final product

The final product is a complete standalone application. The user can import a 3D car model, move around freely in the scene, placing multiple cameras and specific points of interest-icons in and around the car with the functionality of switching cameras when clicking on these points of interest. Edit all settings of the cameras, such as orientation, boundaries for how far you can rotate the camera in each direction, how far or close you can zoom and how the camera will behave in different situations. Setting up different zones in the scene. Creating connection groups for setting all functionality of which camera the end-user has to be in, and in which zone, to view the different points of interest. The user can export and import data. These are the necessary features.

The application also has some extra features such as adding informational text fields and animations of different car parts to points of interest, a searchfield, a mouse-hovering function that highlights selectable items in the scene, transform handles that appears when selecting (clicking on) any object, for easy adjustability of position, rotation and scale. A scene gizmo that shows X-, Y-, Z-axis of the scene. Support for not only exporting all data, but also the possibility to export only chosen categories of objects, or manually selecting each desired object the user wishes to export.

We also made tooltips since there are a lot of settings that can be hard to understand when using the application without experience, and we also implemented a lot of hotkeys.

The application has a preview mode, which is not what in the end goes up on the website since it's only the data that will be exported, but it is a preview only for the user to get a feel of what the web-result will look like.

Future Visions

The biggest vision with this application is to make it as independent from Unity as possible. Now the 3D model of the car still needs some prepping in Unity, to even be able to import the model into the application. To get around this would be major work and not on the agenda before even knowing if the application is better than the previous workflow in Unity, and if it is worth it depending on how often the application will be used.

Some other requests we've had however, that would be something to follow up on, is to be able to import only a selection of chosen objects from a previously exported Json-file, import different environments, import new icons for points of interest and different files for saving and exporting. One file for saving data to continue working on and another for exporting the final data, since we today export all data into one file, but not all data is necessary when delivering the file for the website, but is necessary for importing everything correctly back into the application.

Evaluation and Summary

This project has had a lot of ups and downs, since as mentioned we didn't have any experience of coding in C# or working in Unity. The one person at the company that had most knowledge of the original project in Unity, and had written a lot of its code, quit at the company a few weeks after we started. A lot of our struggles would have been solved faster if we had anyone to ask about the scripts or functionality that were necessary to implement in our application, but hard to understand. Instead we had to debug, test, search online for similar code and try to understand it by ourselves, and we managed to overcome all obstacles.

The first period we worked a lot together, but as our knowledge grew and we got more comfortable and confident about coding, we did more by ourselves. We worked great as a team, followed our planning, and got more done in the end than we thought, and therefore made the application more extensive than the original thought.

One thing that we had a question about during this whole project; who will be the user of this application, was never answered since no one really knew. The tech visualisation team got a request of doing this experience once, and did it directly in Unity, but when they got requests of doing it for multiple more car models they wanted to make a solution so the ones that came with the request

could do it by themselves without having to include the tech team. The application itself was never requested by the ones that will use it.

I collected tons of new knowledges during this projects, mostly coding in C# and working in Unity, including using Unity UI Builder and Shader Graph, but also working through a whole project of developing a new software. Planning, doing a user journey, making a persona, need analysis, working with Git in Sourcetree, working in a team with weekly sprints, testing code, doing user tests and then evaluating important feedback and applying it to the application.

Bibliography

Unity Documentation - https://docs.unity3d.com/ScriptReference/

Unity Forum - https://forum.unity.com/

Volvo Code Library

Tutorials and Inspiration from Youtube:
https://www.youtube.com/c/CodeWithMat
https://www.youtube.com/c/CodeMonkeyUnity

Downloads from Github:
https://github.com/pshtif/RuntimeTransformHandle
https://github.com/yasirkula/UnityRuntimeSceneGizmo