Producing a Level Editor For 3D Environments

CS3D661

Alex Sampson

15015483

Carl Jones

Contents

[Project Summary 3](#_Toc25343406)

[Project Aim 3](#_Toc25343407)

[Project Objectives 3](#_Toc25343408)

[Pre-Existing Level Editors 4](#_Toc25343409)

[Professional Tools 4](#_Toc25343410)

[In-Engine Tools 5](#_Toc25343411)

[Methods of Production 6](#_Toc25343412)

[OpenGL 6](#_Toc25343413)

[Unity Game Engine 6](#_Toc25343414)

[Loading 3D Models 6](#_Toc25343415)

[Selecting Objects 7](#_Toc25343416)

[Project Planning 8](#_Toc25343417)

[References 9](#_Toc25343418)

# Project Summary

## Project Aim

The aim of this project is to create a functioning 3D level editor tool that includes all of the required features able to produce and export a 3D level to a file that is then usable to load the scene.

## Project Objectives

The first objective for the project is to create an environment where 3D models/files can be loaded into. This 3D model will be imported into a local file to ensure the project has no external dependencies, so that it can still function correctly even if the external files are moved or deleted. The Level Editor will have a UI element to display all the available models which will load them from the local files, with a preview version on display. To save on performance, the local version could be either a low-resolution version of the model or a single image file, both options being generated when the model is imported.

The next objective is to be able to access the model file previously imported and position it into the scene. Objects that are placed into the scene must then be able to be selected to enable transformations to them individually. The selection process could be through a UI element which lists every active object in the scene for the user to select from, or directly with the mouse cursor using rays cast in a 3D space from the camera, through the cursors position relative to the world.

The objects, when selected must be allowed to be interacted with, including having the object be able to be translated within the environment by having features to change the objects scale, position and rotation. A secondary objective with the interaction may be to allow the objects colours and textures to be changed, as well as to add any dynamic features such as a constantly moving object, for example a train passing by the scene continuously.

The final primary objective is to export the completed environment to a single file that can be used to load the environment as a standalone file, as opposed to it being confined to the editor.

A secondary objective for the project file management is to implement a save/load feature so that the in-progress levels could be saved inside the editor to be loaded later. This feature isn’t required for the functionality of the level editor, however the ability to save and load the project is a feature that will improve the overall use of the program.

# Pre-Existing Level Editors

## Professional Tools

A picture containing text

Description automatically generatedSource SDK

The Source SDK is a collection of tools used for authoring content for the Source engine. Valve Hammer Editor, included in the Source SDK, is a 3D level creation tool for use with the Source and GoldSource Engines for titles such as Half-Life, Counter Strike and Team Fortress (Valve Corporation, 2019). Hammer is a professional tool used by the official developers as well as publicly available through their Steam platform.

Figure 1 - Hammer Editor

GtkRadiant

Radiant Level Design Tools is a similar tool to Hammer in that it is used to create 3D environments for games powered by the id Tech engines such as Quake, Wolfenstein and Doom (GtkRadiant Team, 2016). GtkRadiant currently exists as an open-source level editor but is also the base of the level editor currently used by Call of Duty developers Treyarch and Infinity Ward as their engine is a modified variant of the id engine.

The tool contains the basic functions of a level editor such as placing objects, lights and entities down to create environments, however it also includes a more advanced toolset for creating set pieces such as triggers, which when supplied with proper key-value pairs, can create a dynamic environment with moving doors, hazards and other interactable objects.

In summary, the two tools referenced above are very refined in their design for a very specific use case, and contain a wide variety of features which allows the users to create intricate, professional levels however this comes at the cost of having a rather high learning curve, as using these features in conjunction with each other can be a burden without the proper experience.

## In-Engine Tools

Halo Forge

Halo’s Forge is an in-engine game mode where players can create and translate objects within the Halo Universe to create variants of pre-existing levels and game-modes. These tools are a lot more rudimentary than the professional tools previously mentioned as they are designed with a focus on user experience and ease of use. Halo’s forge is a mode in which the player can add and remove props from existing levels and place their own down in any configuration within the limitations of the map, as well as the prop count limit that exists for performance and design reasons. The player can switch between a grounded player perspective and a free moving monitor, which allows for instant creation and testing without having to export to an external tool.

Using the editors listed above, both professional and in-engine, helps outline which features are key to this project and how to design the level editor for the specified purpose. To keep the development focused and within schedule, the project will lean more towards the in-engine simplicity to ensure the planned features can be developed fully, with any excess time being used for the more in-depth features more typically found in the professional tools.

# Methods of Production

As the brief outlined above does not require the use of a specific language or technology to be used, the most suitable method must be determined for this project.

## OpenGL

OpenGL is an API (Application Programming Interface) that could be used in this project to handle the graphics and rendering of the level editor’s 3D environments. OpenGL can be integrated into programs written in many different languages, with C and C++ generally being used as OpenGL itself being written in C (OpenGL, 2019).

If the project were to be written in C++ from scratch, OpenGL would be a great tool to use for the rendering, however, this may take a considerable amount of time to fully utilise, leaving less time to develop the key features.

## Unity Game Engine

Unity Game Engine is a multi-platform engine used for a wide variety of productions. Unity contains a vast array of tools that can be used to create games, animations, applications and more. Unity has a wide variety of libraries and components available from the base editor that can be used during the production of this project to easily implement features that otherwise would take longer independently due to the nature of their complexity.

## Loading 3D Models

The first stage of producing the editor is to determine a suitable method of loading the 3D models from a file to be used in the editor. One of the biggest problems to overcome with this aspect is that 3D objects can come in many different file formats such as .OBJ, .FBX and .3DS. All of these file formats will need to either be supported or be stated that they aren’t available in the project as they will all require code that will handle them properly to allow them to be imported into the editor.

The most basic method for handling file formats would be to only include a parser for a single file type, such as a Wavefront OBJ file, and require the user to convert the model outside of the editor which will save time on development for the project however will limit the user functionality as well as the user experience.

Using an existing library such as Assimp, the Open Asset Import Library, would allow any file format supported by the library to be imported into the editor with relative ease, as the library parses the formats and converts them in a uniform manner so that all file formats would appear the same once in the code to be used by the program (Assimp, 2018). The use of Assimp would improve the functionality of the project but has the potential to add an overhead on the time it takes to produce the importer as it would include learning a new library I haven’t previously used. This could also cause issues with the code if not implemented correctly due to inexperience.

## Selecting Objects

When the objects are present in the scene, the editor must have a method of selecting the objects to be transformed. An ideal way to select objects would be to use the mouse and click on the objects in the scene. To achieve this, the program can cast a ray from the position of camera, through the mouse in world position and then select an object if the ray collides with the object. Adding ray casting to the program may increase the development time depending on how it is implemented.

An easier method of selection is to have a UI element that contains a dropdown menu with all of the available objects in the scene to be selected. This method would be a very simple way of implementing selections, however it would hamper the user experience.

# Project Planning

The planning of the project is broken down into two sections, the primary and secondary objectives, with each containing a set of objectives needed. The primary objectives outline the key features needed to produce a simple level editor that hits all the objectives for the project, however, will be lacking in features to create anything more than a basic scene, with very little support for a good user experience. Each section of the primary objectives shouldn’t ideally take more than a week to produce, from conceptual work to implementation, with the exception of the final objective, exporting the finished level, as that will require extra time for research and design.

The Secondary objectives are designed to improve the overall quality of the Level Editor, as well as the user experience of the product by implementing features that are designed to ease the workflow within the tool.

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

Figure 2 - Estimated Time Taken Per Objective

Figure 3 - Time Management Gantt Chart

# References

Assimp, 2018. *The Open Asset Importer Lib.* [Online]   
Available at: http://www.assimp.org/  
[Accessed 22 November 2019].

GtkRadiant Team, 2016. *Radiant Level Design Tools.* [Online]   
Available at: https://icculus.org/gtkradiant/  
[Accessed 18 November 2019].

OpenGL, 2019. *OpenGL API Documentation.* [Online]   
Available at: https://www.opengl.org/documentation/  
[Accessed 21 November 2019].

Valve Corporation, 2019. *Valve Developer Community.* [Online]   
Available at: https://developer.valvesoftware.com/wiki/Valve\_Hammer\_Editor  
[Accessed 29 October 2019].