A level editing process using Autodesk Maya as a base

A. B.
Blekinge Institute of Technology
Karlskrona, Sweden

1 ABSTRACT

A level designing process was developed as part of a game project, where the game used a custom-built game engine. The project used Autodesk Maya as a base for level creation. First, a program that can read FBX files and convert the relevant data to custom level files was programmed. Then, functions for reading these custom files was implemented in the game engine. Tools with a user interface were scripted in Maya for assisting in the level design process. The time needed to export a level was measured and analyzed. Most of the exporting time is taken by the writing and reading of FBX files, thus it may have been a mistake to include FBX files in the process. The prospect to use Maya as a base was evaluated to have been a good idea for purpose of saving development time but has the disadvantages of normally requiring a paid subscription and not providing an exact graphical preview of the finished level.

CCS CONCEPTS

• Human-centered computing~Interaction design~Interaction design process and methods~User interface design

KEYWORDS

Game level editor, level design, user interface, workflow evaluation

2 INTRODUCTION

Level design is often an important part of the game development process. The effectiveness and ingenuity of level authoring tools directly impact the workflow of game content creation. Without applications that can assist in level creation, levels would need to be created through coding and all game objects would need to be placed through the specification of coordinates. This method of level creation is on its own slow to use and test. Thus, level creation tools are very important.

This paper is an evaluation of a game level creation method that was created for a game group project that was part of a third-year collage course. It was created for a specific game with its own game engine; certain aspects of the editor were rushed and partially unfinished. It was designed to be used only by the developers during the course, though it was also designed to be dynamic and creates level files that could be repurposed for other game engines. After the creation of the method, the question is posed: is it a good idea to choose Autodesk Maya as a base for a level creator instead of developing a standalone application?

This paper is structured as follows. Section 3 describes what is expected from a level creator and then explains Autodesk Maya and Autodesk FBX to give context to some pre-existing software used in the project. Section 4 presents the method used, first the development of the editor tools, then the experiment carried out to evaluate its performance. Section 5 presents the result from measuring the editor's effectiveness. In Section 6 the results and development decisions are analyzed and discussed. Section 7 concludes this paper.

3 BACKGROUND

The basic feature that a level editor needs is the ability to create the level geometry and space where a game takes place. For a 3D game; this can mean either that 3D models can be imported, moved, rotated and scaled, or it can mean that level geometry is created directly in the level creation tool. In the games industry, programs such as Maya are widely used to create art assets for games [1]. The game Dreams by Media Molecule includes a game creation tool to create games that doesn't feature polygons at all. [2] The tool allows users to craft surfaces directly.

Autodesk Maya is a 3D graphics program that can be used for 3D modeling, 3D animation, rendering, etc. In Maya it is possible to create tools using scripting, Maya has its own script language called MEL, scripting in Python is also supported which is made more accessible using a third-part library such as one called PyMEL [3]. Maya was chosen as a base for the level editing tools since the game project's developers had prior experience building models, scenes and tools using Maya. In Maya there are several built in tools for creating scenes. There is a hierarchy system for all objects and the ability instantiate models so that the same model is used in multiple locations. These tools and the scripting functionality meant that all necessary functionality could be implemented into Maya, this was considered to be easier and more efficient alternative than creating a level creation application from scratch.

FBX is a file format owned by the company Autodesk. Programs capable of 3D modelling such as Maya and Blender can write FBX files that contain all the data of 3D models, their attributes and their hierarchy in a scene. In an FBX file; attributes can be variables with different datatypes and names. This means that the data in an FBX can be customized to a high degree. The source code for the creation of FBX files is not publicly available, instead the FBX Software Development Kit (SDK) is required to read and write FBX files [4]. Maya has a built in FBX exporter and the FBX SDK was used in the game development project that the level editor was created for prior to work starting on the editor.

4 METHOD

The research methods used are implementation and experimentation. The implementation can be divided into three parts. First, creating a custom file format with level data and a program for writing it. Secondly, implementing functions in the game engine for reading the files and loading levels from them. And lastly, creating tools for Maya to automate as much of the level designing and exporting processes as possible.

4.1 Writing level files

Before the level editing process was started, the game project was utilizing a custom developed program for converting 3D model into a custom file format by reading model data from FBX files. Since there already was a process for model data it was deemed appropriate to implement the level file creation algorithm in the same converting program using a similar approach of reading FBX files and writing custom files optimized for in-game loading by only containing relevant data.

When the game engine was created, its method of handling objects in the game world was inspired by the Entity Component System (ECS) architectural pattern. Technically the system in the engine isn't an ECS because it doesn't implement a number of aspects of it, but it is based on it in the following ways. Every game object in the game consists of an entity with components. The entity on its own has no function other than contain components. The components contain all the game logic and data. There are both graphical components, such as those representing 3D models, and gameplay related components. The file format for a game level is a collection of data that describes all the entities and their components. At the start of the editor's development, the converter program was tested by creating levels featuring only static 3D models. It was programmed to read a scene created in Maya from an FBX file using the FBX SDK, take the relevant data such as positions, and make a list of entities with model components that represent how the level needs to be built in the engine. At first only model components were relevant, but the converter was designed from the start to support multiple types of components.

4.2 Reading level files

When the game engine reads the data, it read how many entities there are, one by one reads the data related to their components, and constructs the level. The component data begins with an integer that is converted to an enumerated type that signifies what type of component it is, then all other component data is read dynamically, it is read as raw data that is then divided up into parameters depending on the type of component.

All models in the Maya scene were by default converted to entities with model components and then given basic box shaped collision in the form of a physics component. But in order to support other types of components a way to represent the entity component structure was devised for Maya. In Maya all models exist in a hierarchy of nodes that can be parental to other nodes. All these nodes can be granted custom attributes which can be of various data types. The converter was programmed to look for empty nodes with names that start with "ent_" and then iterate through its child nodes and treat any model node as a model component and other empty nodes with names that start with "comp_" as other types of components. The custom attributes of those nodes represent the component type as well as all the parameters needed to construct that component in the game engine. This way any variables can be

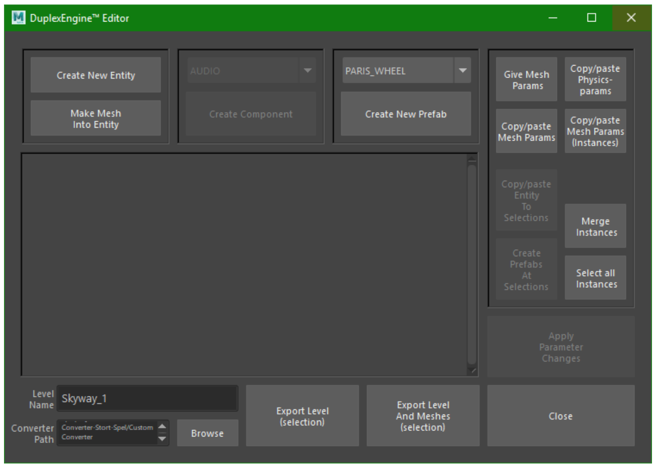

Figure 1: the UI created to interface with the scripts in Maya that automate the level creation process.

defined in Maya and later read by the engine as a component parameter. Any models exported without an entity node as its parent were still treated as an entity with a model component and a physics component with simple box shaped collision.

After components; another type of node called a prefab was implemented. It is a node in Maya whose name begins with "prefab_", it is a way of placing objects that are predefined in the game engine as an entity with a set of specific components. Several commonly used game objects were implemented as available prefabs so that they could be placed more easily and kept consistent. One example of a prefab is the game's collectables, they need to be placed in many places and should have consistent graphical assets and parameters. This also meant that certain components that only had one use case didn't need to be supported as standalone components in the editor.

4.3 Creation of level designing tools

At this point it was possible to completely construct levels in Maya, the only problem was that it is a time-consuming process to create all the necessary nodes and give them the exact required attributes in the correct order. The order of the attributes is required to be in the same order each time so that the data can be read in the correct order in engine. To solve this problem, a number of scripts and a user interface (UI) was created to automate all the required setup in Maya. The scripts were written in Python using PyMEL and the UI was designed using the program QT Designer, the UI can be seen in figure 1. The first step was adding buttons that set up nodes in Maya that represent entities, components and prefabs. The button for adding components and the associated drop-down menu can only be used when an entity is selected. All nodes created using these buttons have all the required parameters and default values. The buttons in the rightmost square in figure 1 activate scripts that are intended to help with level creation. They were not planned before development started on the editor but were added during the designing of a level as ways to assist with frequently preformed tasks.

The large empty space is figure 1 was intended to dynamically display all the parameters of selected entity in the scene. As it stands, to change parameters, each component has to be selected one by one. Displaying and allowing editing of all components of an entity could be more intuitive than using Maya's menu for custom attributes. This feature was considered inessential and a low priority during development and was not completed as other matters were prioritized.

The converter program is programmed to read all the FBX files that are placed within a specific folder, one folder for levels and one folder for models, and convert them into custom file formats. The user interface features two buttons for exporting a level, one to place the level in the levels' folder, and one to export it both to the levels folder and the models' folder. This distinction was made so that the model conversion can be skipped to save time if no models in the level were added or modified. The converter program also has a function where it will after the initial conversion continuously check for new files and changes in the existing files to see if something needs to be converted again. This way the converter can be left running and is not required to be reopened every time a new file needs to be converted. This is intended to optimize the workflow for exporting and converting levels. The workflow for exporting a level to the game engine is as follows. First all relevant objects need to be selected in Maya, this can be accomplished by putting all relevant nodes in a group and selected that group. Secondly a directory path to the converter program needs to be entered, this only needs to be done once and is saved as a variable even after Maya is closed, therefore it can be skipped on subsequent exports. Thirdly, an export button is pressed, and Maya begins to write an FBX file. If the level is to be exported into both the levels' folder and models' folder, the script copies the file in the levels' folder and pastes in into the models' folder. Finally, if the converter program is running it observes that new FBX files were added and will begin converting them. The converter places the finished files directly in the game engine directory. This directory is hard coded into the converter program. The game engine was programmed with a feature to reload a level from a file with a button press, thus if the converter was running prior to the export; changes to a level can be loaded with a total of two button presses. One button press in the Maya UI and one in the engine.

4.4 Measuring the performance of level conversion

To test the performance of the workflow, the time needed to export a sample level was measured. This was done by capturing a screen recording of the process from when the export button is pressed to the moment that the converter program displayed a message saying the conversion was done. This recording was then opened in a video editing program, the exact frames of the button-press and the converter message were marked and the time period between the two was noted. This was done both for an export of only level data as well as an export with level and model conversion. The level in question was a complete level created with the editor as a part of the game project which the editor process was made for. It was observed that the portions of the exporting process that seemed to take the most time was the export of the FBX file by Maya and the reading of the FBX file in the converter before the converting started. It was hypothesized that the FBX writing and reading process could be sped up by deleting unused data in the Maya scene. One prevalent type of unused data was the unintentional custom attributes that many of the models in the scene had. These unintentional attributes were assigned when the models were imported into Maya from a different modeling program called Blender. Thus, a script was created to clean all the models in the scene of any irrelevant attributes. And the same time measurements were carried out to be compared to pre-cleanup measurements.

	Before cleanup	After cleanup
Level data	68 s	35 s
Level and model data	71 s	40 s

Table 1, results from the time measurements of the level export process

5 RESULTS

The resulting times can be seen in table 1. The level that was measured consisted of 1679 nodes, 1197 of these were 3D models and the remaining 482 were empty nodes used to denote entities, non-model components, prefabs, and nodes simply used to group together other nodes for organizational proposes. These times were recorded on a PC with an Intel® Core™ i5-8600K Processor. Other programs were running in the background during the converting; however, the CPU usage was monitored and did not reach 100% at any point.

6 ANALYSIS AND DISCUSSION

6.1 Effectiveness of level conversion

The resulting time measurements in table 1 are sample times from a level with an amount of complexity the editor process was intended for. The time that converting takes is very likely dependent on the level of complexity of the level, the amount of polygons in the models and the hardware of the computer used to run the converting. As such different scenarios would likely yield completely different results. It is observed that the most time-consuming part of the conversion is the writing and reading of FBX files. Since the code for FBX files is not publicly available it is not known how those algorithms scale with the amount of data in the level. After the Maya scene was cleaned of irrelevant attributes the times for conversion were nearly halved, even so, the conversion process was slower than initially hoped. As such it can be stated that it was a mistake to device a process that relies on FBX files for reading data from a Maya scene. Possible alternatives would be to write a level file using a method that can interface with a Maya scene more directly, such as a python script or a C++ plugin. This would mean relying less on unknown code and would likely make irrelevant data in the scene less of a factor since it could be programmed to only access relevant data. As such cleaning up irrelevant data wouldn't be necessary. A disadvantage of not using a program that reads FBX files is that using FBX files written by other programs than Maya no longer is an option. Though the level conversion process already relies so heavily on the tools made in Maya that converting files from other programs isn't a very useful option.

6.2 Maya as a base for a level editor

The decision to use Maya as a base for creating game levels was made primarily because of time constraints. There was prerequisite knowledge about scripting in Maya and it features extensive tools for scene building. The main purpose of the editor process was to provide a way for level content to be created for a recently developed game engine in a school project. This purpose was achieved before its deadline and allowed a large amount of content to be featured in the finished level compared to if the level had to be constructed through code. However, when compared to editors used as tools for creating commercial products there are several disadvantages in using Maya as a base. First of all, Maya requires a paid subscription to be used to create content for commercial products. This can be a major downside for any project that wouldn't otherwise utilize Maya. Another disadvantage is that Maya doesn't offer a live preview of how the level will be rendered in the game engine. Which can be an issue for artists placing decorative assets and also doesn't allow for live measuring of graphical performance. This can be remedied by creating a plugin for Maya that streams data into the engine so that the game application itself becomes a preview. This method could potentially make the level conversion program obsolete as the game engine then potentially would be a better candidate for writing level data files. Furthermore, at the start of the project, the process for converting a level for testing in the game engine was not expected to take more than half a minute. The ability to load changes takes only two button presses, which was planned from the start, though applications that allow editing and running the game seamlessly definitely have an advantage as the level does not need to be converted or loaded.

At the start of development, a big advantage of using Maya seemed to be the ability to create or edit models whist editing the level without the need to manage and import files. While this is a convenient feature; many game engines already allow the creation or editing of assets [5].

7 CONCLUSION AND FUTURE WORK

To conclude, the level editor process served its function to allow level content to be designed in Maya and be used in a game engine to an extent that was compatible with all features that were necessary. The time necessary for importing a level into the game engine was longer than expected which breaks up the level creation workflow more than what was desired. Although, the ability of being able to load a level after adjustments in two button presses is very convenient. Since time was an issue and the methods used were easy to plan out, using Maya as a level editor was a good idea. If time hadn't been an issue, the results of developing a standalone application or level editing mode for the game engine has potential to make for a better end result. Partly because it would not require a Maya subscription and partly because it would provide a live graphical preview.

As mentioned, one way to further development the process would be to create a plugin for Maya that allows data to be streamed to the game engine so that the game application can provide a live preview of the level with the same shaders and correct rendering of the game. This could potentially solve one of the biggest disadvantages of this method, which is the lack of an accurate graphical preview.

Furthermore, the user interface of the Maya tools was left unfinished, for the method to be used in future projects it could be beneficial to finish the entity parameter changing section. Another possible feature to add would be to add the ability to support more components without changing any source code, this would make level editing accessible to individuals without programming experience.

Another way to expand this method for use in other projects would be to create a library for reading level files and redesign the level reading to be more dynamic in a way that could benefit other game engines.

REFERENCES

[1] T. Wyeld and D. Hobbs, "Visualising Human Motion: a First Principles Approach using Vicon data in Maya," 2016 20th International Conference Information Visualisation (IV), Lisbon, Portugal, 2016, pp. 216-222, doi: 10.1109/IV.2016.83.

[2] Nebelong, Martin. "Create a landscape in Dreams for PS4" 3D Artist, 2019, pp. 46-52.

[3] Chad Dombrova, PyMEL for Maya, help.autodesk.com/cloudhelp/2018/JPN/Maya-Tech-Docs/PyMel/index.html, (accessed on 11 March 2021)

[4] Autodesk, What is Autodesk FBX Technology, help.autodesk.com/view/FBX/2020/ENU/?guid=FBX_Developer_Help_welcome_to_the_fbx_sdk_what_is_autodesk_fbx_technology_html, (accessed on 2 March 2021)

[5] V. B. Vasudevamurt and A. Uskov, "Serious game engines: Analysis and applications," 2015 IEEE International Conference on Electro/Information Technology (EIT), Dekalb, IL, USA, 2015, pp. 440-445, doi: 10.1109/EIT.2015.7293381