Bachelor's thesis

Information Technology

Game Development

October 2017

Mohammad Kafami Khorasani

GENERATING AN IDEAL FEATURE LIST AND RECOMMENDING SOFTWARE FOR VIDEO GAME PROCEDURAL URBAN MODELLING

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Game Development

October 2017 | 42 pages

Supervisor: Werner Ravyse

Mohammad Kafami Khorasani

GENERATING AN IDEAL FEATURE LIST AND RECOMMENDING SOFTWARE FOR VIDEO GAME PROCEDURAL URBAN MODELLING

The recent developments in computer systems' processing technology allow graphic artists to create increasingly complex 2D and 3D video games assets. Consequently, the process of generating 3D graphics for video game is becoming more complicated. The purpose of this study is to recommend a list of effective features that accelerate and ease the workflow of 3D urban modelling, and help environment artists to create high-quality and optimized outcomes.

To obtain the desired list of features, the fundamentals of video games environment design were reviewed and the feasibility and advantages of relevant methods were studied. Then, the process of custom urban assets modelling consisting of designing and creating several types of environmental elements was described. At this stage, the applications and software, such as 3D studio Max 2015 and GhostTown 1.0 plugin were examined. Afterwards, two broadly used procedural 3D generating applications, namely Esri CityEngine and Houdini were technically inspected, and their capabilities and respective feature sets were reviewed. As a result, a final feature list was compiled.

The generated feature list provides information on essential capabilities of procedural 3D generating applications in handling a series of consecutive procedures such as: geospatial data entry, road and street networks, parceling, building constructions, texture mapping and finally optimization of the outcomes. Based on the features list, a table was also generated to benchmark the investigated applications.

Despite the user-friendly interface and strong node based façade generator, GhostTown does not deliver game-oriented results and requires major time and effort for optimization of geometry and texture map. Therefore, this plugin is not recommended. Considering the advantages and limitations of CityEngine and Houdini and their complementary quality, a hybrid method is proposed in this thesis. Since CityEngine includes a strong toolset for managing the geospatial data entry, terrain and road network generation, parceling, rule-based building modelling, it can favorably be employed at the beginning phases. While Houdini can collaborate partially in parceling and building generating, it can also perfectly fulfill the rest of the procedure, consisting of models' topological perfection, optimized texture mapping and creation of the level of details.

KEYWORDS:

Video-Games, Procedural Modelling, Procedural generation, Architectural environment, Houdini, Unity, 3Ds Max, Esri, CityEngine, Ghost Town, VIRDI.

CONTENT

FIGURES

TABLES

LIST OF ABBREVIATIONS (OR) SYMBOLS

3D	3 dimensional
CGA	computer Generated Architecture
CPU	Central Processing Unit
FPS	Frame Per Second
GIS	Geographic Information System
GPU	Graphics Processing Unit
L-system	an approach to represent a polygon by thin version of its shape
LOD	Level-Of-Detail, an approach to create different polycount flexible alternative 3D models to improve the efficiency of rendering
Mental Ray	a Rendering software by Mental Images, Berlin, Germany
NURBS	Non-Uniform Rational Basis Spline
NVIDIA	an American graphical technology company, Santa Clara, California
PC	Personal Computer
QGIS	a free and open source geographic information system
UDIM	U-Dimension, a method to identify integer blocks in UV space
UV	U and V stand for 2D texture axes in 3D graphics
VIRDI	Virtual Driving Inspection Video Game
VR	Virtual Reality

1 INTRODUCTION

Working as an architectural environment designer in the game industry, not only requires architectural designing knowledge and a sense of aesthetics, but also an understanding of some quick and efficient methods to provide speed and quality. Use of a fast feature is very important when creating a massive and high detailed game environment like a cityscape, since using old fashion methods to create a huge number of details is either impossible or infinitely time consuming. Besides, the quality and efficiency of the modelling's result should be deeply considered, otherwise it causes a hampered gameplay.

Modelling of architectural structures is processed with different approaches being "procedural generation", "digital scanning" and "manual modelling", while the latter approach is not always efficient neither affordable for the volume used in Video-Games.

The purpose of this thesis is to provide an ideal list of features that environment artists should refer to when selecting a modelling tool. Cityscape modelling involves different stages including topography of the terrain, roadmap, constructions, basic geometry beside a variety of belongings such as structural elements (e.g. beams, poles, concrete platforms), window frames, rails etc. Moreover, natural objects commonly existed in city urbans, roads, pavements, street signs, traffic lights … need to be included.

The thesis commences with some related basic theories and continues with description of the above-mentioned approaches regarding their applicability and advantages. Then, the thesis proceeds to explain different requirements and limitations in Video Games on different platforms concerning efficiency, polycount and other measurements.

The practical part of the thesis consists of a development journal describing the modelling process of custom urban assets for a video game environment. The process comprises measuring and creating several types of semi-official and housing constructions using currently prevalent applications and software as below:

- 3D studio Max (The main 3D modelling environment)

- Ghost Town 1.0 (procedural building generator)

- Unity (Game engine)

The study then focuses on technical testing of two alternative plugins/stand-alone applications (procedural architectural model generators) in the market, evaluates their capability and compares their respective feature sets with the method and application which was journaled. This phase leads to a reflective section where a final feature wish list will be compiled.

2 MODELLING AND TEXTURE MAPPING FOR VIDEO GAMES

3D modelling is the process of creating a 3-dimensional graphical representation of a real world or imaginary object or shape using computer software. A 3D model's structure is made from few main elements (Figure 1), such as:

Vertex (pl. Vertices): The end points where two or more lines meet and create a corner.

Edge: A segment on the boundary of a surface or intersection of two faces in a polyhedron geometry.

Polygon: A planar surface bounded by a group of segments (edges) is a polygon. "Each segment is joined to the next at its endpoint, and the last line segment connects back to the first" (Eck, 2016).

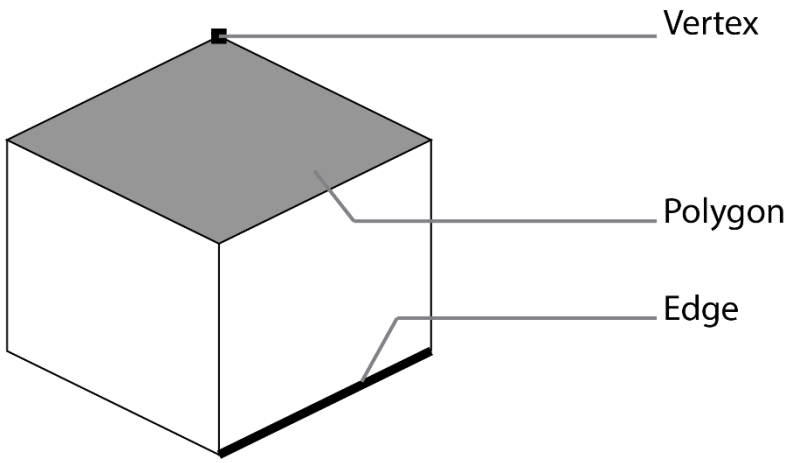


Figure 1 Elements of a 3D model

2.1 Principles of 3D modelling in Video Games

To enhance the efficiency, a video game 3D artist should be concerned about modelling references, performance limits, model simplification, topological perfection of a model and texture mapping optimization. These considerations cover all possible modelling fields including environment, objects, characters and others. To expose an astonishing game appearance, a designer should first determine the exact time and location in which the story of the game happens. Perhaps in a special part of the history? Or in the future? Does it fictionalize a scenario in an unknown region of our planet earth? Is it going to be visualized in a populated modern city urban or deeply in a natural environment with no manmade construction? Nevertheless, also having a strong imaginative background, a good step to bring the imaginary portrayals to reality is to prepare clear references.

Next level is to consider the factors that influence the efficiency of the model implemented within a game environment. Several limitations exist in video game performance regarding the targeted gaming platform capabilities and also the genre and style of the game itself. Software-based technical limitations depend on the game engine, graphical rendering pipeline, light models, mapping technology, collision models, etc., whereas the hardware technical limitations are basically the memory limit (texture memory), processing units, input devices and interfaces, connectivity and bandwidth, display resolution and the type of platform. Gameplay limits refer to game-play time and speed, camera angle and viewport distance, size and scale of models, use of the static or dynamic 3D models, among others.

One of the most important challenges that artists usually face in Video Games' environment design, is the processing restrictions, where a fundamental concern is the polygon count. Since 3D modelling techniques commonly used in Video Games are polygon based, the process often requires a dense mesh to generate more realistic models. High detailed meshes include many polygons, each with their own datasets. These polygons are sent to the CPU and GPU once at a time when a model is displayed on the screen. Therefore, the more polygons a model has, the longer it takes to process. In other words, 3D modelers are constantly faced with the trade-off between performance and aesthetics.

In order to preserve a feasible dataset to render, 3D models are restricted in their number of details. A game environment composed of lower poly assets will render more smoothly

and the result will be a more effective frame to frame real-time render. A standard gaming system, in essence, is a rendering machine which is designated for constant rendering of each frame at 30 FPS frame rate. In case, the frame rate drops during the game, the result is a frustrating gameplay. This applies to all types of gaming platforms, such as mobiles and tablets, game consoles and PCs.

The required efficiency span is different from one system to another. The reason is the range of details that the target platform can support. Compared to console games, PCs typically have more processing power to run higher resolution models with higher range of details. Mobiles and tablet games require extremely efficient execution since rendering capability of mobile platforms is considerably inadequate for complex visualization.

In movie 3D modelling, not everything needs to be built for the pre-rendered model. Usually film production 3D artists only create those elements of the setting which audience can already see on the screen. In the game environment, it is necessary to make most of the things viewable from 360 degrees. Considering that the player always walks and surfs through the game level, different sides of the 3D assets become uncovered and exposed. Compared to the Video Games, in a movie, if the camera never moves behind a set, the back part does not require to be built. Though this is definitely true for aspects of the gaming world that a player can't practically get to. (Dargie, 2007.)

The topological perfection and optimized texture mapping will be discussed further in the section 2.4 entitled 'Optimization concerns'.

2.2 3D modelling technical approaches

There are 3 common technical paths for modelling architectural space including manual modelling, digital scanning and procedural generating. Regarding the objective of this thesis, some relevant contents are discussed below.

2.2.1 Manual modelling

The first and most usual modelling method is the manual modelling, when the artist or engineer creates, shapes and polishes the model him/herself using a computer application. Manual modelling consists of different approaches such as subdivision modelling which is one of the two main polygonal modelling techniques. In this approach,

the modeler starts with a primitive geometry such as a box, then shapes it in a desired way. The other polygonal modelling technique is edge modelling in which the model starts being created based on a loop of polygons. Another technique called NURBS/Spline modelling stands for Non-Uniform Rational Basis Spline modelling which is technically based on generating smooth meshes lofted between various Bezier curves.

Finally, in the digital sculpting the modeler works with organic meshes and very probably with huge number of polygons in form of high-resolution meshes. This method is considerably fast since it enables the artist to use a graphical tablet and its digital pen to engrave, mold and form the object exactly as in sculpting in clay. This approach is an effective style for creation of digital characters, organic objects, sculptures, also different modules of architectural structures (Common Modeling Techniques for Film and Games, 2017)

2.2.2 3D Scanning

Another well-known 3D modelling path is digital scanning. In this approach the real-world models are scanned and measured into a computer (Digitizing) by various types of scanners to provide astonishingly high-level photorealistic assets. Scanning encompasses different sets of techniques and variety of 3D scanning styles and methods with different applicability like the Structured light scanning, Photogrammetry and Laser scanning technology. (Common Modeling Techniques for Film and Games, 2017)

2.2.3 Procedural 3D modelling

Procedural modelling is a technical approach in which the objects, foliage, modules, textures etc., are being generated automatically based on a set of parameters and algorithms. This technology is used for large-numbered sets of modular structures and foliage. Regarding the huge variety and complexity of these polyhedrons, it can be boundlessly time-consuming to create them manually. An example of this is simulating broad number of buildings in an urban/cityscape as a video game environment (Figure 2). Therefore, programmers tackled this difficulty by using code and algorithms to automatically generate the desired shapes. These algorithms can take advantage of a pre-designed module library or a set of functions that work with primitive geometry and textures. (Ebert et al., 1998)

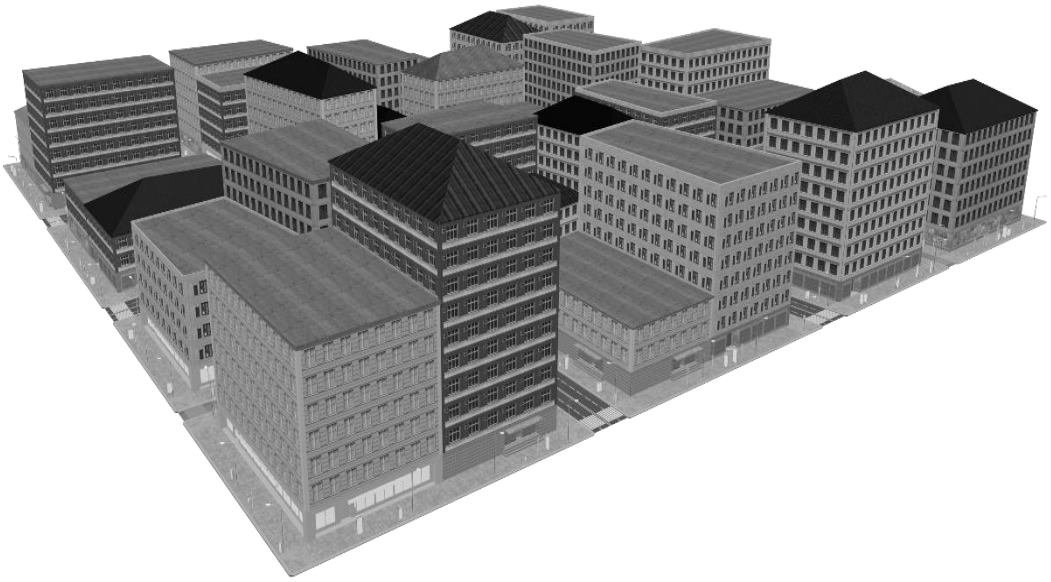


Figure 2 A simple urban model using procedural technology

> "The key property of procedural generation is that it describes the entity, be it geometry, texture or effect, in terms of a sequence of generation instructions rather than as a static block of data. The instructions can then be called on when required to create instances of the asset and the description can be parametrized to allow the generation of instances with varying characteristics. A typical example of this approach would be the population of a forest with procedurally generated unique trees." (Kelly & McCable, 2006)

Nowadays, a number of graphical applications work with procedural generators, so artists can benefit from the aspects of this technology along with their visual techniques. Examples of computer application that work with procedural modelling are SpeedTree by Interactive Data Visualization and CityEngine by Esri R&D Center Zurich.

2.3 Texture in 3D graphics

The digital portrayal of a surfaces that has 2 or/and 3-dimensional qualities such as color, brightness, transparency, opacity, glossiness, reflection, etc., is called texture. The term ‘texture map’ refers to a 2-dimensional image that applies to 3D model’s polygon faces to implement the above-mentioned qualities. A texture map may be a bitmap image or a procedural texture.

2.3.1 UV and UVW mapping?

In 3D graphics, while the X and Y denote the 3D object’s axes in model space, U and V stand for 2D texture’s axes. The process of creating texture map for a 3D object is called UV mapping. It is a process whereby you create, edit, and otherwise arrange the UVs that appear as a flattened, two-dimensional representation of the surface mesh, over top of the two-dimensional image to be used as a texture as it appears in the UV Texture Editor. (Mullen, 2009)

UVW mapping stands for a 3-dimensional technique for coordinate mapping. In contrast to UV mapping which is R3to R2, UVW is R2 to R3, to paint a surface based on a solid texture. The 3rd dimension enables the texture map to wrap in complex ways onto irregular surfaces. (Mullen, 2009)

Texture maps are created to correspond to the UV coordinates of 3D model’s surfaces. In form of bitmap images, texture maps are either captured from real-life photos, or manually painter in a computer graphics application such as Photoshop. Procedural texture maps are developed by triggering an algorithm instead of manual creation using actually stored data. The advantage of using procedural textures is low storage cost, unlimited resolution and easy mapping.

Procedural textures are applied to model a volumetric portrayal of natural material like wood, concrete, stone, etc. Using fractal noise and turbulence functions as a numerical representation of the “randomness” found in nature, illustrates a natural look onto the rendered result. (Eck, 2016)

Different elements of a texture map (Figure 3) are described below:

Color (or Diffuse map): The most important role of a texture map is to add color to a model's surfaces.

Specular maps: A specular map (or gloss map) is a grayscale image that addresses those parts of a model that are supposed to be shiny or glossy. It also measures the magnitude of the shininess.

Bump, Normal map: Normal map is a type of texture map that projects a more realistic indication of bump or indent on a face of a 3D model.

Reflection map: These texture maps tell the software which surfaces or parts of the 3D model are designed to be reflective. A reflection map is needed whether the object is only partly reflective, or if the level of reflectivity is non-uniform.

Transparency map: Transparency map functions exactly like Reflection maps, except it addresses the portions that must be transparent. It is often used for the surfaces which are too computationally expensive to duplicate, like bushes, tree leaves, fences, chains, etc. Transparency map is highly applicable for natural objects and also architectural constructions in game environments. (Lifewire, 2017)

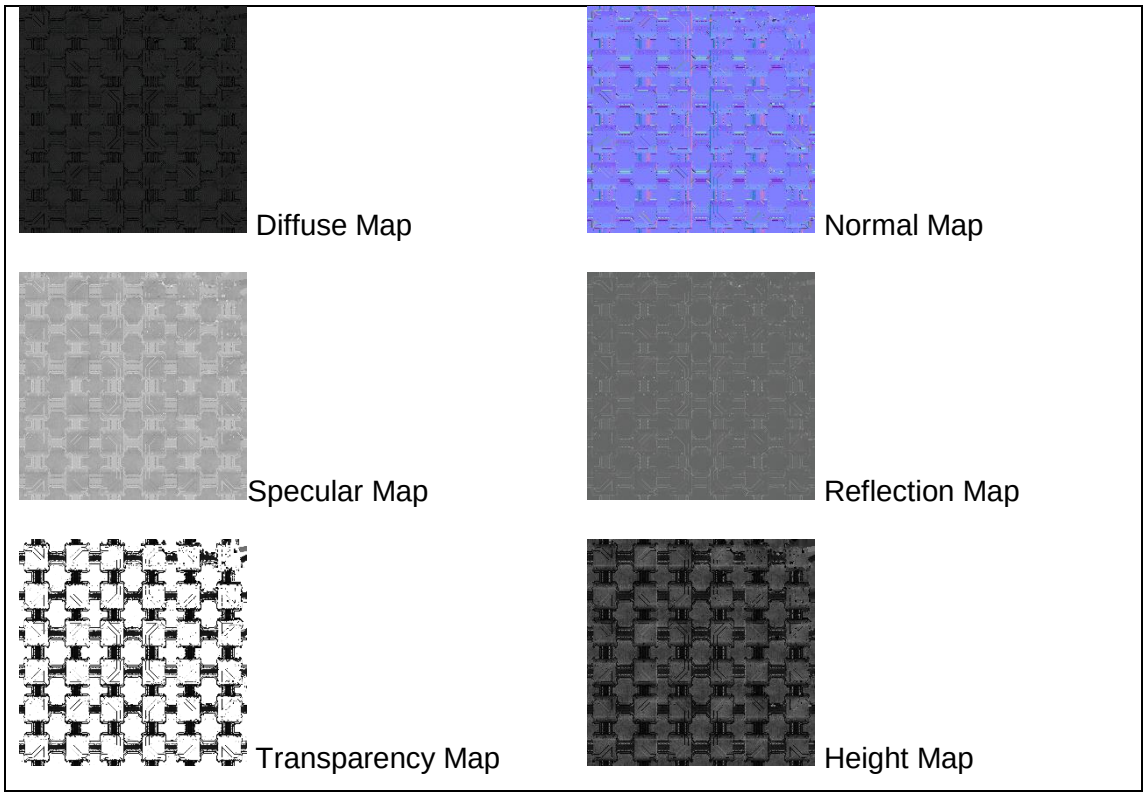


Figure 3 Graphical representation of texture maps

2.4 Optimization concerns

There are a set of items that must be optimized to produce a highly efficient model for video game. By improving these items and models, we ensure that the quality of work is guaranteed both in the geometry modelling and the texture mapping process. Some of these items are as follows.

- Polygon count reduction

The most important concern for 3D game artists is the number of polygons generated in the assets which must be processed in rendering. Artists apply variety of techniques to their models, not only to lower the polygon count, but also to retain its aesthetical state.

Since the higher polygon number requires the rendering system to spend more time and memory for processing, minimum number of polygons is crucial for smoother and faster frame to frame rendering.

- Topological perfection

A very common fault that occurs during the modelling procedure, especially in architectural models, is geometry imperfection. Regarding the variety of compound shapes in architectural objects, polygon distortions, fractures and holes are very possible to happen. Creating a flawless 3D geometry requires accurate refining of the model geometry, preferably before the texture mapping.

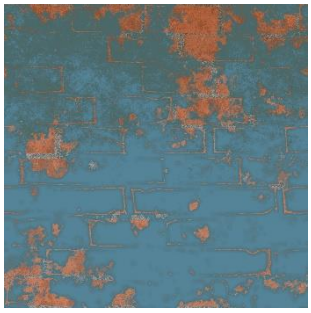
- UV maps and texture correction:

Consistent, clearly unfolded and stitched UV maps that have been unwrapped and spread decently upon the texture image can shorten the process and provide a smooth and efficient real-time rendering. Moreover, UV overlaps and interferences are preferred to be fixed in order to have a correct light and shading.

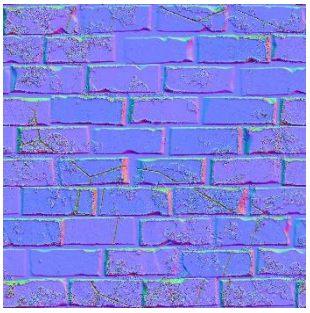
Additionally, there are advanced methods and processes to create more complex models at a lower cost. Bellow we discuss briefly two of these advantageous techniques.

2.4.1 Applying the Normal maps

Normal maps enable us to add details such as bumps to surfaces without increasing the polygons. They also replace the surface normal in which the artist can bake a pre-modeled complex geometry of two to three million polygons down to a normal map that retains those high-resolution model's component space data. Then a streamlined model that emulates a more basic proportion of the complex model is created in significantly lower polygon number (e.g. two to three thousand). In next step, the normal map data is applied to this low-resolution rendition of the original high-resolution model. The result immediately appears like a silhouette of our complex monster, though with highly affordable rendering value (Dargie, 2007) (Figure 4).



Defuse map



Normal map

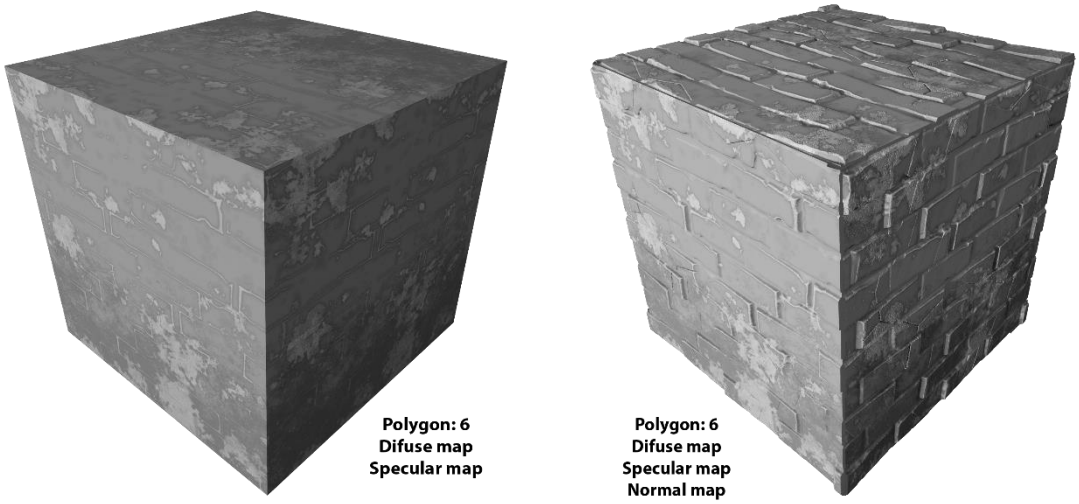


Figure 4 Adding normal map renders a more complex result without any changes in the original geometry.

2.4.2 Level-Of-Details technique (LOD)

A common practice to avoid rendering the unnecessary high-resolution elements is to create Level-Of-Detail models (Figure 5). In a game, when the player walks up to some building construction from the far end of a long road, chances are those are not consistent models the entire journey for the building and its supplements. When they are far away, lower resolution models and textures are used. It's because the details cannot be recognized at that length, so it is not necessary to use CPU time to render those more complex elements. As the player approaches towards the building, there may be two or three changes that swap the model and textures out with increasingly higher resolution assets, until it appears right up to the camera. If done properly, these "swap outs" go unnoticed for the most part. (Dargie, 2007)

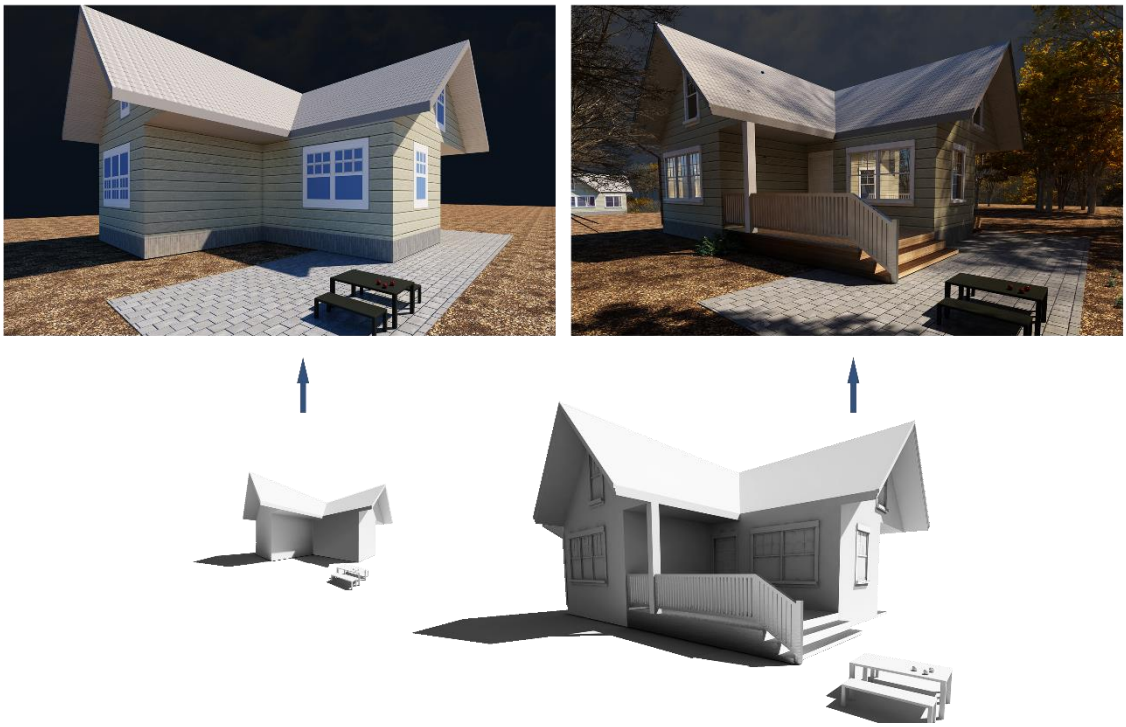


Figure 5 Level-Of-Detail. High resolution model (2136 polygons) on the right-hand side and Low-resolution models for further distances on the left-hand side (348 polygons)

2.5 3D studio Max and Ghost Town 1.0 review

Ghost Town is a procedural generator plugin for 3Ds Max, appropriate for emulating the urban environments (Figure 6). This parametric script can build a massive city with only few clicks. Ghost Town features enable us to create many short and tall buildings and other low- poly and high-poly objects in addition to parametric texture map in a significantly quick way. This plugin works with algorithms, parameters and rules and provides the artists with brilliant options such as road layouts and terrain, proper road and street models, parametric texturing tool etc. (Figure 7).

Figure 6 City model, Generated in Ghost Town, adjusted in 3Ds Max. (William Ehrendreich, 2016)

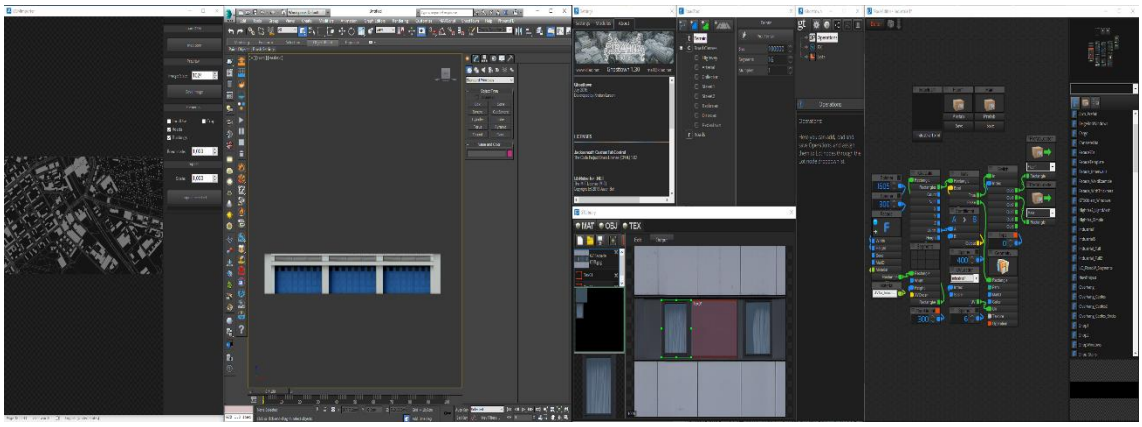


Figure 7 Ghost Town and 3Ds Max user interface

3 DATA AND METHODS

3.1 environment for VIRtual Driving Inspection

VIRDI (Virtual Driving Inspection) is a VR video game as an alternative for the conventional driving inspection methods. In this newly developed program, the player as an applicant for driving license can drive a car in Virtual Reality environment using simulator devices consisting pedals, steering wheel etc. Despite the logical game actions and realistic driving process implemented in programming algorithms, the environment design (Figure 8) of the project has been processed with a life-like appearance. Bellow, the workflow of the construction modelling is discussed in detail.

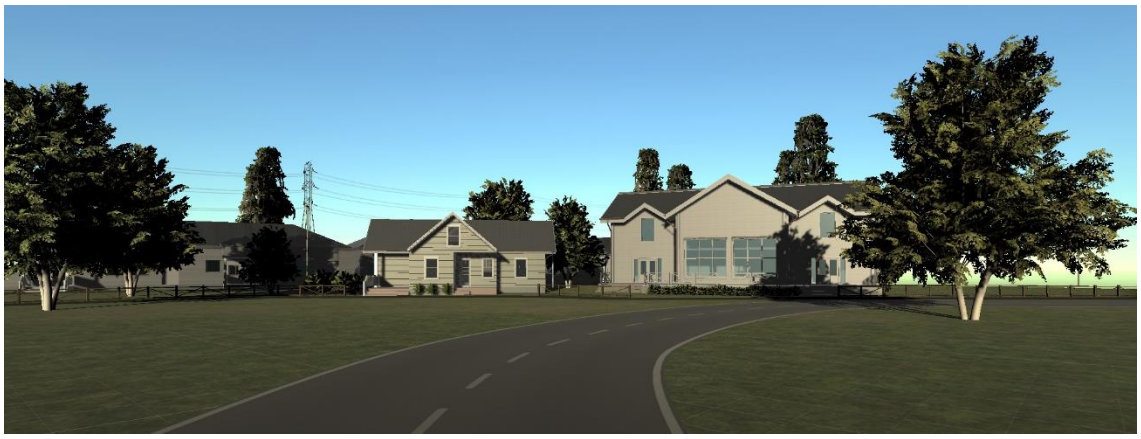


Figure 8 Virtual Driving Inspection environment scene

3.1.1 Workflow

Since a simulation of a Finnish suburb area was required in the project VIRDI, the procedural generator feature needed to work with a set of customized facades, streets, exterior objects etc., organized in our desired scenery. Therefore, a list of custom-built assets, their details and textures is created and stored in a fitting library.

Modelling of the different types of building constructions consists of several stages that vary from one building type to another, regarding the use of geometry, facets, textures and specially the repetitive or non-repetitive details such as windows, engravings, doors, balconies etc., however, the general process follows regular levels of creation which will be elaborated further.

3.1.2 Facade library

- Building up a set of simple prefabs in Ghost Town

In order to generate the building facades, another level of assets such as openings and wall objects must be designed and collected in the prefabs list used by the facade library of GhostTown, consists of several entrance doors, Windows, Panel sheets, Stairs, Entrance vestibule, Terrace, Balcony etc.

Different styles of doors and windows, from classical wooden to modern sliding metal types are created to support variety of facades for different building styles. These prefabs are also customizable as either single or multiple windows/doors, having different frame thickness, riser or top shade's shape and size etc.

- The Building's Facade structure

Now it's being described in detail that how the first façade in the library has been generated using Ghost Town node Editor tool, also how different segments have been employed on the wall's surface, finding their final formation by being connected to suitable prefabs in the library such as openings, risers, lattices etc., accordingly.

Step by step creation of the façade starts with building up a wall's outline applying a Façade node and adjusting the dimensions in GhostTown node editor. Next step is to divide the plane wall into different partitions according to our designed blue print, imaginary picture etc. GhostTown node editor provided us with several options that do this division for us, such as Sidebar modifier, Top/Bottom modifier, Vertical/Horizontal Segment, Segment modifier and so on. It is also possible to push in or pull out some parts of the wall to build a more complex shape.

After we modified different partitions in a desired way, it's time to insert our wall objects into some of these partitions. We can apply a single or multiple window, bathroom windows, doors, balcony etc. For the rest of the wall that has no objects and openings, there are some other applicable prefabs such as panels, bumps, curves etc., or there can also be a simple texture only. Now it would be great to level out every detail on our wall objects. (Figure 9)

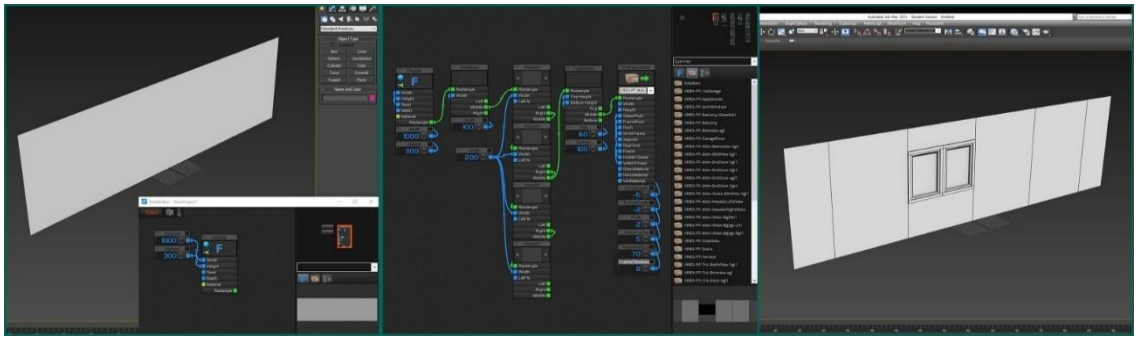


Figure 9 Ghost Town node editor. From Left to right, Facade modifier, Prefabs modifiers, Final result

Every customizable parameter, previously implemented onto prefabs are available to adjust and change shape and dimension by just clicking on the number visible on the prefab's icon. By applying our customized assets beside GhostTown's flexible adjustments such as random generator in node editor, many versions of each façade, contrasting in measurements and structure can be modeled.

- Investigating the random generator, measuring and adjustments

A very good idea that can speed up the process is to use the Random generator in node editor and apply it onto segment modifier instead of manually partitioning the facades. This will enable us to implement same façade in various places and receive a different result every time. The reasoning is that while segment modifier divides the wall surface into (N) identical segments, the Random generator (with range of M), which is connected to (P) different prefabs via it's input wires, randomly generates diversified objects on segments and obtains numerous probable structures with proportional dimensions. (Figure 10)

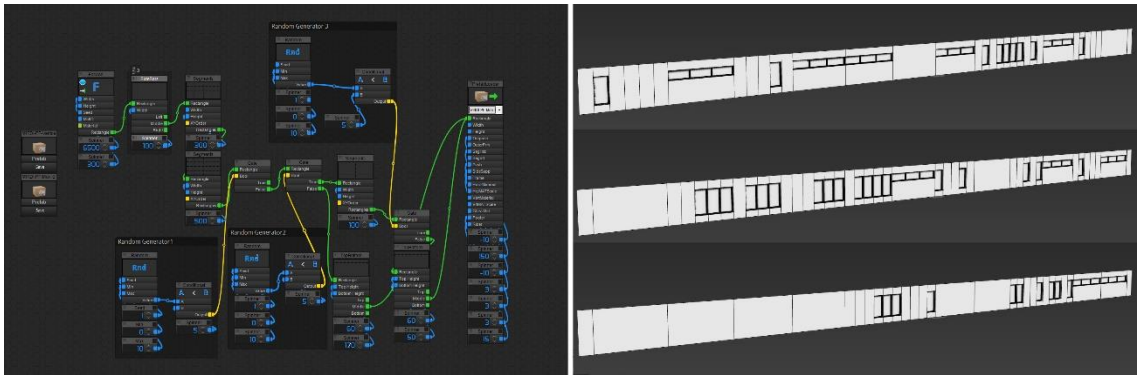


Figure 10 The same node format on the left top, has given us plenty of different facades compositions.

3.1.3 Texturing process

Now that we created and stored our prefabs in the Ghost Town library and used random generator or manual techniques to complete our facade structures, assigning a set of exquisite textures to facades can prepare an authentic library. To accomplish this important, a sufficient texture library needs to be organized.

- Designing a unique texture, using GhostTown texture editor

Using GhostTown texture editor, enables us to make our own texture out of favorite slices extracted from actual bitmap images. After cropping texture sections, Texture editor transfers and places them in a unique frame. These slices of textures can be organized depending on their scale, coordination, vertical or horizontal expansion etc., according to artists preferences. Subsequently, two files, one in XML file format and the other in bitmap image common file formats such as Jpg, png or bmp will be produced and stored. The bitmap file is the image reference of the texture map while the XML file functions as semi-programmable file that addresses different sections on the image file. Next step here is to provide Ghost Town plugin with a texture material in the material editor. A standard material can be suitable for this purpose. The bitmap image file is being assigned to the material map and added to Ghost Town material library.

Now by assigning the mentioned material to the facade node in GhostTown node editor, it’s going to appear on the model in the view port. subsequently, a UV operator calling the XML file, is needed in the node editor to make the material functional. For each portion of the facade, there must be a Geometry operator in node editor that calls the relevant segment in the bitmap through its UV loader input. This UV loader genuinely includes two entries itself. One entry is a number to set the segment address while the other entry sets the size of the UV map upon the polygon.

3.1.4 Generating the buildings

In the real-life architectural construction, not every side of a building have the same look except in some constructions with a symmetrical architectural plan. One problem with the Ghost Town would be the identical façades that are automatically generated for every side of a building geometry. To solve this problem, one useful method could be to primarily generate the rough building models on our area map, then pick our preferable facade from the library and apply it onto the selected wall of the building, considering that every single building must have a unique group of side views (Figure 11).

Figure 11 Iteration of the identical facades on every side of the building geometry

This step can be a little time consuming, because we may not like random generator to distort our buildings unsightly, hence beside completely using random generation, we would rather do many adjustments or creations manually in between.

E.g. applying facades that include random generation operator on an entire side view of a building generates different shapes for each floor on one side of the building, whereas commonly they should look the same or at list follow a certain formation. In this case it's better to apply the facade group only onto one floor of a story building, export the building from GhostTown editor to 3Ds max as an editable mesh, then clown the first floor onto all the other stories of the building geometry manually. (Figure 12)

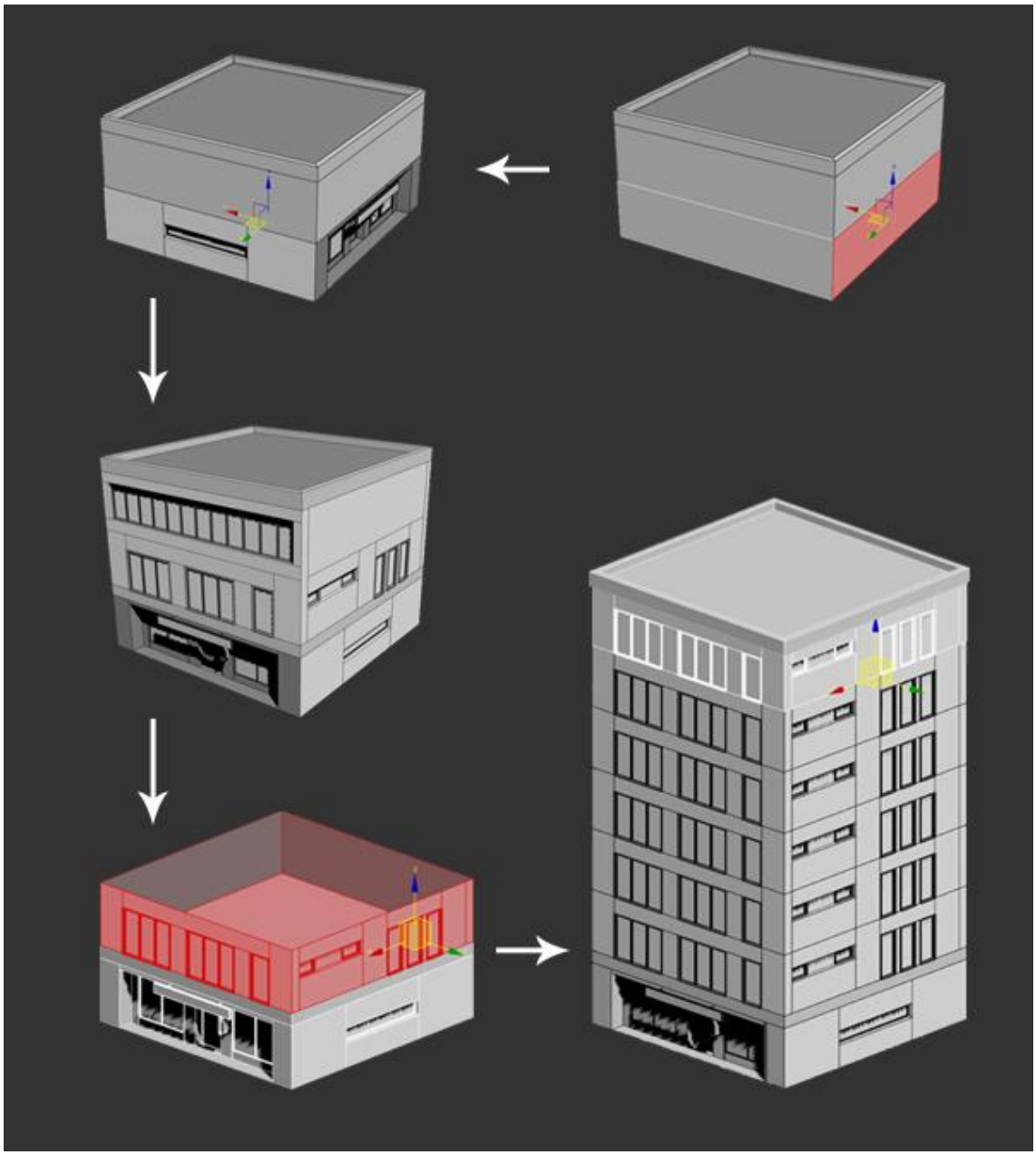


Figure 12 Different steps of applying customized facades on a buildings surfaces

- Building types

The building construction group considered for the project VIRDI comprised three main types of Detached houses, Row houses and Story buildings. Depending on the building type, the process of modelling and texturing may alter slightly.

- Approaches to build different types

Despite the certain workflow for all of building architectural styles in general, there are considerable steps that vary from one type to another. Regularly, a box shaped or a more complex geometry with designated XYZ dimensions is generated primarily, using GhostTown building operation (Depending on building type, floor number varies). In the next step, façades are being applied on a certain facet (except top and bottom) of the building geometry.

3.1.5 Challenges and difficulties with GhostTown

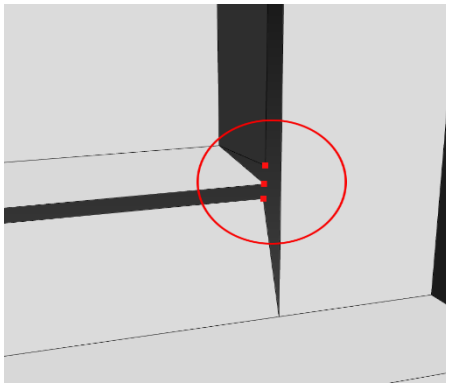
Though Ghost Town as a strong procedural 3D generating tool allows the artists to benefit from spectacular features that accelerate and simplify the process of 3D modelling it may result few shortages that must be taken into consideration.

First and foremost, as long as the Ghost Town is designed to work with rectangular and cubical modules, organizing these modules beside each other most of the time leads to detached polygons and discrete vertices that don’t have pair on the adjacent polygon’s edge. Regarding the scale of the game scene and the surroundings, welding of the distinct sub-objects and optimization of the geometry can consume a boundless amount of time and effort.

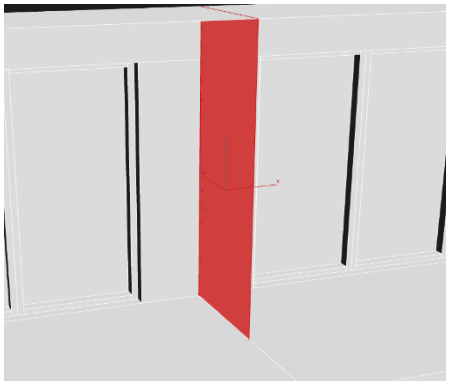
Secondly, numerous hidden polygons are likely to be generated in complex models following the object transformation on facades in which the main geometry’s dimension is not an exact multiple of its segment prefabs.

The other fault is related to texture maps. Using the GhostTown texture editor for parametric texture maps which address the texture section to repetitive polygon segments causes overlapped UVs. These Overlapping UVs cause a distorted shading and the result is a chaotic reflections and highlights on the object’s surfaces.

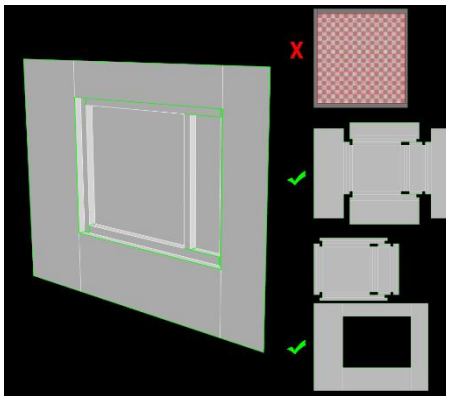
The latter problem can be refined and fixed by baking the textures onto the models, but it still needs manual unwrapping in a new UV channel which is an infinite load of work for a massive cityscape model. Aside from spectacular advantages of Ghost Town that makes it highly suitable for pre-rendered Media results, architectural emulation etc., for Video Games it still needs further development for more efficient features (Figure 13).



Detached vertices in the mesh



Spare polygons caused by creating bump/groove



Overlapping UV maps created in Ghost Town (upper image) vs unwrapped UVs properly spread over the texture (two lower images)

Figure 13 Common faults that may occur in Ghost Town result mesh

3.1.6 The final Construction models

- Refining the Polygon count on extracted meshes from ghost town

After extracting the building mesh modified in GhostTown plugin, it needs optimization, refining, also a little bit of other adjustments. As it has been discussed in previous chapters, the Ghost Town generated models commonly have extra polygons. Also, tetrahedron based modelling in GhostTown creates detached vertices that must be welded, stirred or removed afterwards and this will considerably affect the polygon count.

Now that the structure of the geometry is corrected and finalized, the texture maps are ready to be optimized. Primarily, those parts of the texture map that have been stretched or distorted along the corrected geometry must be refined and rescaled in the UV editor.

In case the geometry corrections are done carelessly and regardless of the consequences on the texture maps, it might cause an extremely time-consuming correction process. To prevent that, it's suggested to avoid moving the edges, also "clean" (Ctrl + Backspace) the vertices instead of deleting them. After repairing, the texture map is ready for optimizing the UV unwrap.

- Modifying the UV MAPS for texture baking

In this level, 2 layers of UVW-unwrap modifiers are being applied on the object. The first modifier fixes and keeps the current Texture map and the UV coordinates on the initial channel Whereas the second layer provides a new channel where the new texture map structure is going to be modified on. Primarily model's UV maps are being placed on the texture bitmap with an appropriate coordination and scale, using Unfolding and Flattening techniques in the UV editor. Next, all of the joints and connections will be attached and optimized by applying the Stitch and Weld techniques. Finally, the entire UV map will be packed inside 0 to 1 coordinate.

- Texture Baking

The new UV map that has been polished and optimized in the second UV-unwrap layer is ready to replace the old UV map. For this purpose, primarily the Texture map needs to be baked onto the object through the second channel. The resolution, type of the texture map, UV channel, relevant Bitmap file, scale … are some of the options that should be adjusted in the texture baking editor window.

After baking, the old UV map layer is being deleted while the newly visualized one with a unique layout needs to be transferred to the initial channel. Afterwards, in material editor, the new texture bitmap that is rendered and saved in the texture library, will be assigned to the object. As the last step, all the modifiers will collapse down to the polygonal object.

Texture bitmaps are suggested to be checked after test rendering, as they may still have few quality issues such as bleeding on the borders, stretching etc. Polishing and improving the quality of the textures in a photo editor application like Adobe Photoshop can be much effective at this stage.

- Level-Of-Details

When an object in the game scene is far away from the camera, many details cannot be noticed. An optimization technique called Level-Of-Detail (LOD) is applied to reduce the number of triangles rendered for an object as its distance increases.

- Making Level-Of-Details for buildings

In LOD technique, several versions from one model is being generated, differing from each other by polygon count and amount of details. As the object goes further from the camera, the existing model is replaced by the lower detailed one. In VIRDI project, for every building, there are two models generated (LOD0 is the full detailed and LOD1 with much fewer details, sufficient for the long distance) which is enough to meet the requirements.

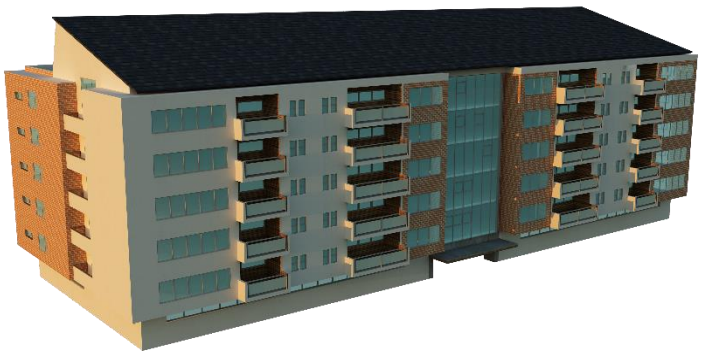
- Editing textures for LOD models

Since many details are removed from the LOD1 object’s geometry and replaced by plain polygons and surfaces. A new bitmap image replaces the original one. This image contains the same textures for existing old surfaces, also covers those new polygons and any other missing areas for the recent object model.

3.1.7 Project results (ongoing)

Bellow, A list of buildings that are modelled out of custom facades collection, using GhostTown node editor and texture editor is displayed in a table. Quantity of the quadratic polygons and vertices in addition to number of texture maps and their quality obtained by 3Ds Max's 'Baking to texture' technique is mentioned bellow each model. (Figure 14 and 15)

Considering rendering efficiency which may result in a smooth game play and impressive frame rate or oppositely a frustrating real-time rendering, not only poly-count is a key element but also quantity, resolution and optimization of texture maps can also affect the rendering procedure in a similar way. So, it is very important to lower the number of textures maps and improve balance between precision and resolution of bitmaps.

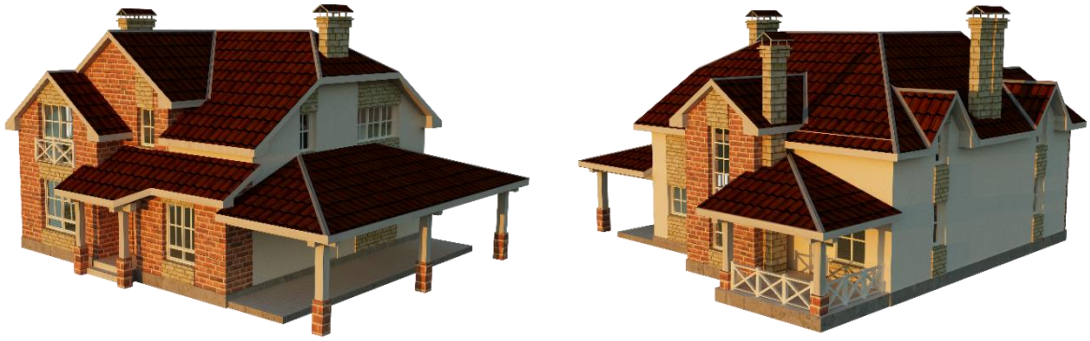


Polygons: 28379, Vertices: 23769, Texture resolution: 2K, Texture bitmaps: 6

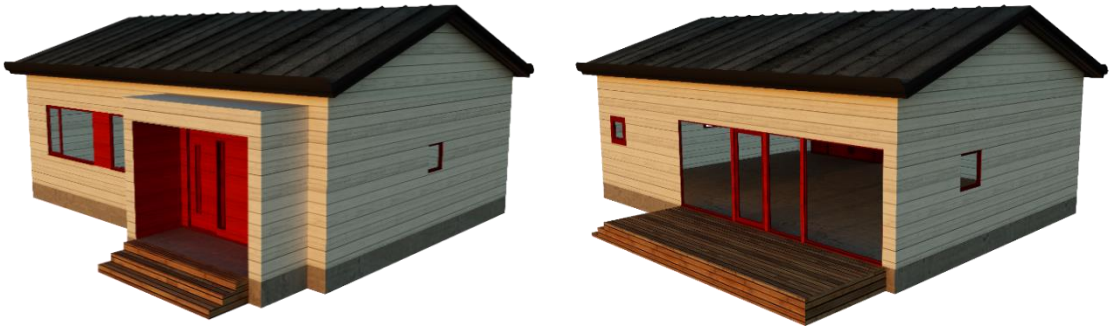
Polygons: 2854, Vertices: 1988, Texture resolution: 2K, Texture bitmaps: 2

Polygons: 242, Vertices: 143, Texture resolution: 2K, Texture bitmaps: 2

Polygons: 2648, Vertices: 1649, Texture resolution: 2K, Texture bitmaps: 3



Polygons: 6000, Vertices: 4473, Texture resolution: 2K, Texture bitmaps: 3



Polygons: 1173, Vertices: 822, Texture resolution: 2K, Texture bitmaps: 2

Polygons: 4065, Vertices: 2727, Texture resolution: 2K, Texture bitmaps: 3

Polygons: 1121, Vertices: 730, Texture resolution: 2K, Texture bitmaps: 2

Polygons: 4315, Vertices: 2138, Texture resolution: 2K, Texture bitmaps: 2

Polygons: XXX, Vertices: XXX, Texture resolution: 2K, Texture bitmaps: 2

Polygons: 1370, Vertices: 768, Texture resolution: 2K, Texture bitmaps: 2

Polygons: 9324, Vertices: 5500, Texture resolution: 2K, Texture bitmaps: 3

Polygons: 8246, Vertices: 6725, Texture resolution: 2K, Texture bitmaps: 3

Polygons: 12769, Vertices: 6953, Texture resolution: 2K, Texture bitmaps: 4

Figure 14 Result building models and their quality parameters

Figure 15 VIRDI game Scene screen shot

3.2 Testing similar features and application

3.2.1 Esri CityEngine

“Esri CityEngine is a stand-alone software product that provides professional users in architecture, urban planning, entertainment, simulation, GIS, and general 3D content production with a unique conceptual design and modeling solution for the efficient creation of 3D cities and buildings.” (arcgis.com, 2017) (Figure 16)

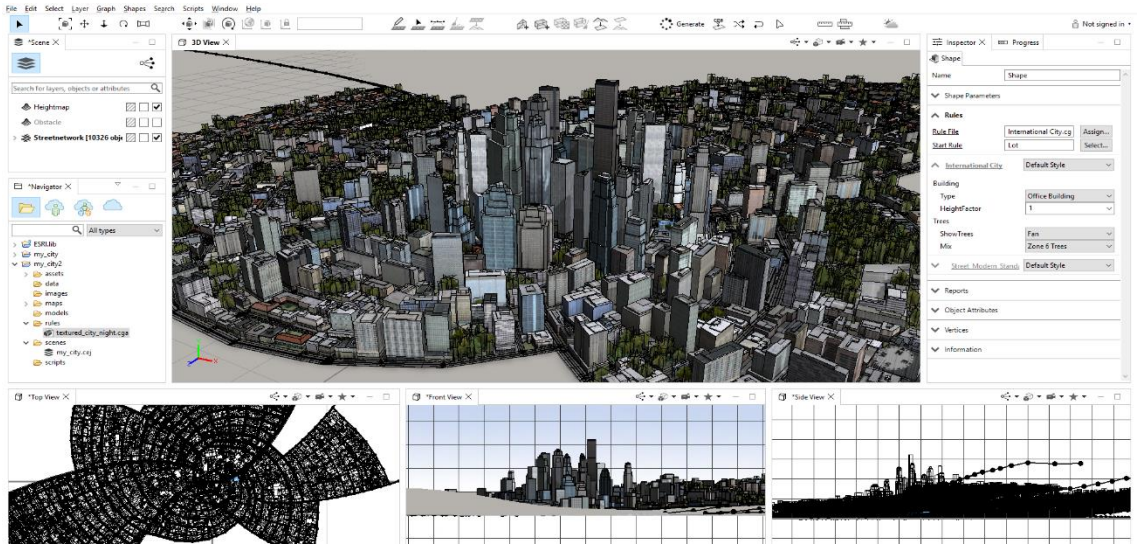


Figure 16 CityEngine’s User Interface

This program allows users to efficiently create 3D cities based on their existing GIS data. (Figure 17)

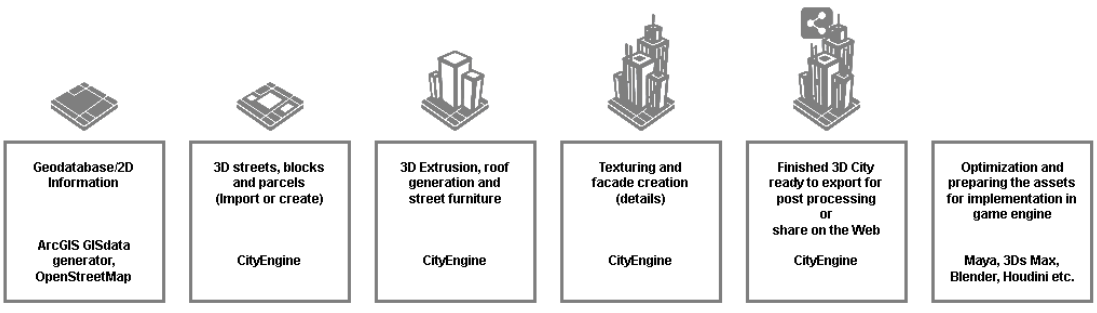


Figure 17 Workflow to create 3D cities from existing 2D/3D GIS data (Esri.com, 2017)

Key topics such as geometry data, featured attributes and procedural rules provide the 3D content with great complexity and lifelike precision as the volume of details used in each topic can be desirably arranged. (arcgis.com, 2017)

- CityEngine's Toolset and features

GIS/CAD Data Support: This software supports geospatial data formats such as Esri shapefile, KML, GDB, OSM etc., in order to import/export any vector data and geospatial information such as building footprints with ideal specifications or line data to generate street networks. CityEngine works with industry-standard 3D formats such as Collada®, Autodesk® FBX, DXF,3Ds, Wavefront OBJ which can be exported out to RIB (Pixar's RenderMan®) and NVIDIA's mental ray® MI format. (arcgis.com, 2017) (Figure 18)

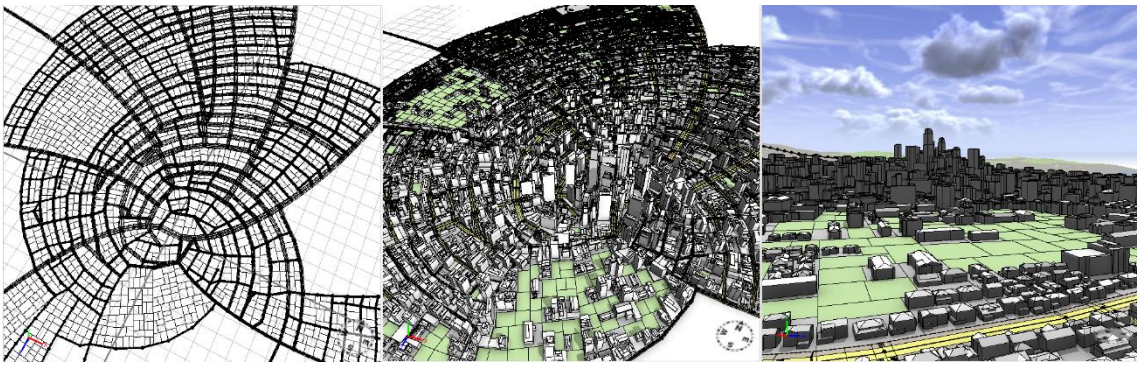


Figure 18 Road intersection, City blocks and extrusions in CityEngine based on Geospatial Data (rfx.com, 2017)

Parametric Modeling Interface: In CityEngine, a user or artist can interactively adjust specific parameters for buildings and streets, for instance the construction's height and width can be adjusted according to the CGA rules. (rfx.com, 2017)

Street Networks Patterns: This standalone software offers a street map toolset to design and construct urban layouts efficiently. In addition to topography of the terrain, a set of street patterns as grid, organic or circular are available. (rfx.com, 2017)

Reporting (BIM for Cities): Rule-based reports are generated and updated to analyze urban design by calculating the quantity of Floor Area Ratio and Gross Floor Area. (rfx.com, 2017)

Rule-based Modeling Core: The 3D modules generated by CityEngine are rule-driven. These Computer Generated Architectural (CGA) rules allow user to control geometry assets, texturing, mass and the proportions on a city scale (Figure 19). Objects

generated based on CGA rules can easily be iterated through several design algorithms (arcgis.com, 2017; rfx.com,2017)

Figure 19 Rule based 3D modelling

Dynamic City Layouts: To design and edit the city layouts including streets, blocks and parcels, CityEngine has provided a toolset which allows the user to control the construction or block subdivision via parametric interfaces. (rfx.com, 2017)

Facade Wizard: This wizard creates rules from textured model or an image map, to provide size-independent Level-Of-Detail and extendable facades modules. (rfx.com, 2017) (Figure 20)

Figure 20 Creating rules from image or textured models (Desktop.arcgis.com, 2017)

3.2.2 Houdini engine

Houdini is a procedural 3D generating application developed by Side Effects software in Toronto. The origin of this application is traced back to Prisms, an application developed in 1980's. During the past years, the developer team has made an extensive effort towards making the software more beneficial and convenient (Figure 21) resulted in the latest version's innovative user interface and features development. Houdini can be used either independently or in collaboration with an existing workflow. (Sidefx.com, 2017)

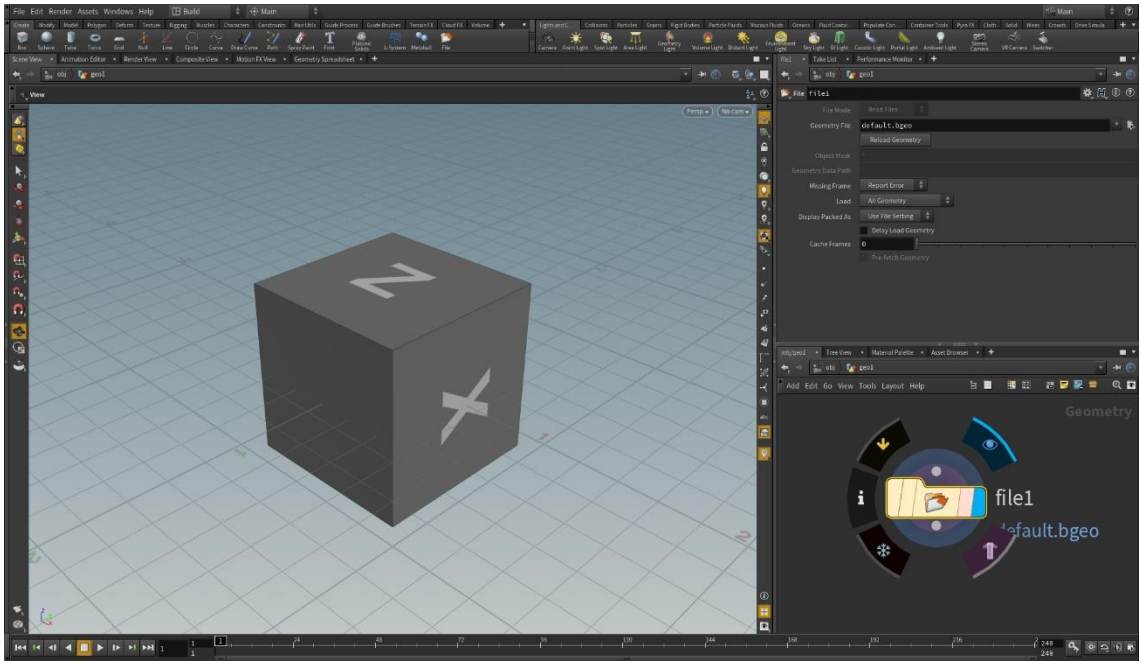


Figure 21 Houdini FX interface

Houdini Engine is a plugin used for data integration between Houdini and other 3D modelling applications or game engines. For game makers who work with programs such as Cinema4D, Autodesk Maya and 3Ds Max, Unity and Unreal Engine, employing Houdini Engine enables importing Houdini assets within these applications. This plugin processes the node network in these assets in the background, as the result is delivered to the host application's viewport. (Sidefx.com, 2017)

Latest versions of Houdini (15 and later) include strong toolsets that are particularly useful for building procedural assets to be used in game editors. These toolsets include

texture baking techniques and support world-space normal maps, U-Dimension[1] textures, UV overlapping and UV mesh boundaries in the viewport. As well, this toolset allows developers to import and export their normal maps and easily convert displacements and bump maps to normal maps. Moreover, PolyExpand2D geometry node helps to figure out the straight-skeleton[2] and generate road network from connected polylines, by building an offset geometry at a measured space from a graph of linked polylines. (Sidefx.com, 2017)

There are two strong geometry nodes in this application that identify the efficiency of the result's geometry. These nodes named Curvesect and Fuse, respectively find the intersections of different polygons or points of minimum distance between them, and merge, or split those vertices. This can be a suitable solution for the geometry imperfections which were previously addressed as one of the Ghost Town flaws.

Various mechanisms in Houdini are determined for a wide range of game 3D modelling and production procedures. However, since this study is focused on architectural procedural modeling, the relevant tools and options of this broadly used software will be investigated.

Houdini procedural generating creates high quality 3D models using a sort of L-System, Lindenmayer, which is a set of symbols, formal grammar, and a parallel rewriting system, introduced in 1968 by Aristid Lindenmayer. L-Systems utilizes an alphabet of symbols for generating a collection of rules to expand the strings parallelly to larger ones for describing the development of branching structures (Hanan, 1992). Furthermore, this system enables interpreting the generated strings to create either realistic or schematic images of the geometric structure, using computer graphics techniques (Hanan, 1992).

The workflow of procedural city generating with Houdini is more or less like other node based alternatives commonly in use. Houdini's features are very efficient, particularly for game developers who want to generate optimized procedural assets to export to game engines.

Modelling of cityscapes and architectural environments using procedural techniques in Houdini is feasible through using topology and road map either generated inside Houdini or obtained from initial GIS data imported from other applications such as

[1] UDIM (U-Dimension) is a method to identify integer blocks in UV space by creating a linear number (Seymour, 2014). (https://www.fxguide.com/featured/udim-uv-mapping/)
[2] An approach to represent a polygon by thin version of its shape that is equidistant to its edges.

OpenStreetMap.com. Accordingly, the artist would be able to employ this data using the Esri CityEngine or a free and Open Source GIS tool named QGIS (Qgis.org, 2017).

The GIS data prepared with QGIS in SHP or DXF file formats can be imported to Houdini to generate parcels and plots and create attributes from image maps. Additional courses like distance calculation and raster data attribute creation are possible to be processed in QGIS to have the data ready for 3D modelling back in Houdini viewport (Figure 22).

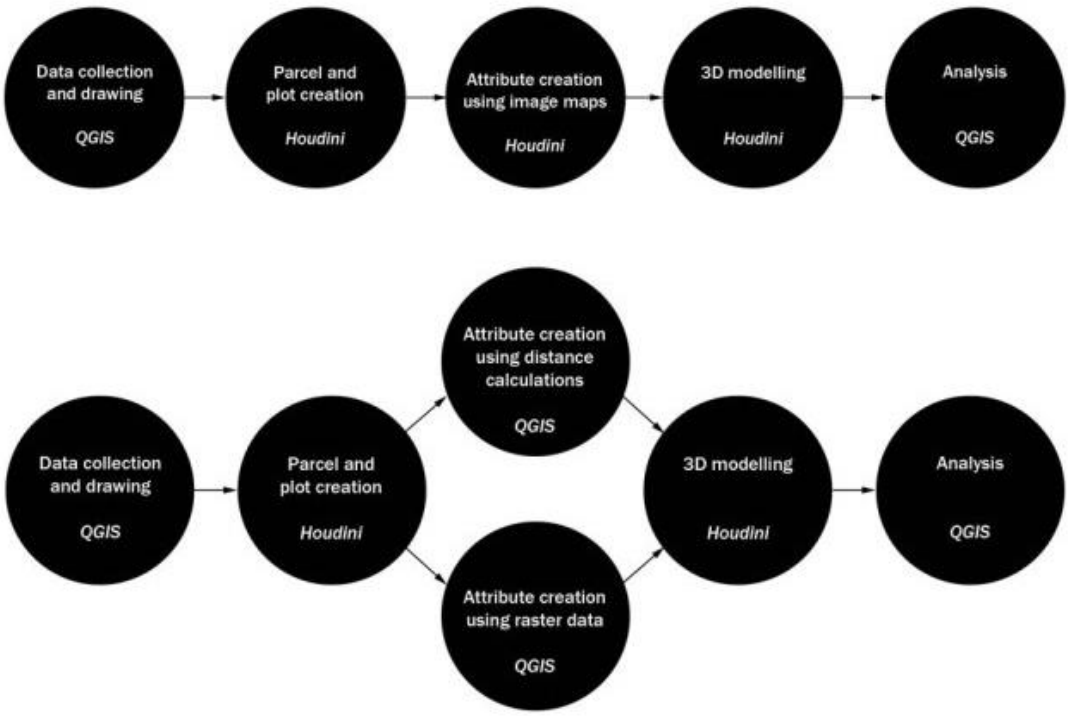


Figure 22 QGIS-Houdini two workflow standards (GIS4Design, 2017)

Using the GIS data beside employing a combination of CityEngine, 3D modelling application such as 3Ds Max and Houdini standalone programs can also be an effective method. GIS data entry, streets and building generation, urban parcels and blocks can be measured and created by CGA rules in CityEngine. These assets are then opened in 3Ds Max for further refining in geometry and textures. The result as .OBJ file is imported to Houdini, in order to create a texture path for 3D objects via its developed features involving Python programming language. Over and above that, other objects existed in environments like cars, trees, people etc., can be generated and instantiated using Houdini crowd simulation (Figure 23).

Figure 23 CityEngine - Houdini Workflow

Topology and roadmaps can also be designed and created from scratch in a unique style using Houdini's toolset such as PolyExpand2D geometry node (Figure 24). There can be many custom approaches to start a Video Game environment modelling in this feature-rich[3] application. Therefore, 3D graphics artists can use the combination of tools and methods to generate their environment procedurally, use Python programming to import GIS data to Houdini viewport or even use it as isolated application to build the game world on their own designed terrain, imaginary street network, building styles and facades.

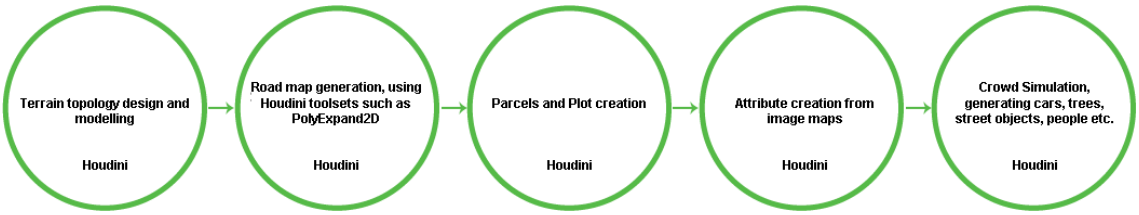


Figure 24 Workflow levels for urban modelling, using Houdini Independently

[3] Application that provides strong options and functional capabilities available to the user

4 DISCUSSION AND CONCLUSIONS

4.1 List of remarkable features for qualification of toolsets

This study's review and workflow experiments of the above-mentioned procedural 3D generation tools (Ghost Town, CityEngine and Houdini) resulted in establishing a desirable list of features that should be considered when selecting urban modelling software that will result in efficient buildings and other cityscape asset generation:

- Map data and GIS data Support
- Fast and easy Procedural modelling interface
- Parametric tools for dynamic terrain and roadmap
- Rule based mass modelling for building generation
- User-friendly viewport
- Customizable façade generator
- Parametric texture editor
- Level-Of-Details modelling capability
- Attribute creation capability based on image maps
- Static asset input
- Development Kits or plugins for procedural modelling inside other applications
- Scripts and coding
- UV map quality
- Results geometry optimization
- Collaboration with other tools in an existing workflow
- Independent usage
- Game orientation
- Affordable price
- Tutorial, learning kits or documentation
- License types

As this study implied previously, these applications are capable of being used independently or in association with a combination of other tools. However, to obtain criteria for a decent comparison, the listed features covered by each tool need to be investigated. Table 1 shows which of the software packages in question meet the desired list provided above.

Table 1 Reviewed applications and their associated List of features

	Features		Ghost Town	CityEngine	Houdini
1	Map, GIS data Support		☑ OSM	☑ FGDB, DFX, OSM, SHP	☑ DFX, SHP
2	Modelling interface type		Procedural Node-based	Procedural Adjustment panel, Scripting	Procedural, node-based, Associates with Python
3	Parametric tools for dynamic terrain and roadmap		☑ Terrain & Road generator, Lot division, Street, pavement	☑ Design/edit urban layouts, streets, blocks and parcels	☑ PolyExpand2D, Curvesect, Fuse etc.
4	Rule based mass modelling (Buildings)		☑ Build Operations, Main control panel, .XML	☑ CGA Shape Grammar Language	☑ Node editor
5	User friendly viewport		☑	☑	☑
6	Customizable façade generator		☑ Node editor, Facade library, facade details	☑ Rule based generator, Facade wizard	☑ Node editor, Modification panels
7	Parametric texture editor		☑ Based on image maps, XML reference	☑ Based on image maps Texture library	☑ Based on image maps, Texturing tools
8	Level-Of-Details support		☑	☑	☑
9	Attribute creation based on image maps		☑ Texture editor	☑ Façade Wizard	☑
10	Static model import		☑ Object library	☑ Drag and Drop	☑ Regular asset import
11	Plugin, SDK		☑ Essentially a plugin	☑ CityEngine SDK	☑ Houdini Engine plugin
12	Scripts and coding		☑ Max script	☑ Python	☑ Python
13	UV map modification		Poor optimization	Medium efficiency	High, Optimized UV maps
14	Results geometry optimization compatibility		☑ pre-rendered media ☑ Architecture ☒ Real-time rendering	☑ pre-rendered media ☑ Architecture ☑ Real-time rendering	☑ pre-rendered media ☑ Architecture ☑ Real-time rendering
15	Collaboration with other tools in a workflow		☑ Geospatial data generators, 3Ds Max	☑ GIS data generators (ArcGIS, OSM…)	☑ GIS data generators (QGIS…)
16	Independent usage		☒ PLUGIN	☑ Architectural design ☒ Video Games	☑ Manual Topography ☒ GIS data import
17	Game oriented		☒	☒	☑
18	Price		Low (90-100 USD)	Moderate	High (4,495 USD Annually)
19	Tutorial, Learning kits or documentation		☑ Few basic Tutorial available in official website Kilad.net	☑ Basic tutorial available in Esri Website. ☑ Courses available online	☑ Basic tutorial available in SideFX.com. ☑ Courses available online
20	License	Trial		☑ Trial	☑ Apprentice
		Single user		☑ Node-locked	☑ Houdini FX
		Multiple user	☑ Premium	☑ Network	☑ Educational

According to the Table 1, GhostTown has a user-friendly interface and strong node based façade generator and is capable of handling map data and creating attributes based on actual image maps. However, it does not deliver game-oriented result and demands major time and effort for optimization of geometry and texture map.

CityEngine, as a standalone software, has an advanced and at the same time user-friendly interface. It supports a wide range of file formats for geospatial data entry and includes efficient terrain and road map creator toolset, rule-based parceling and building generator and parametric texture mapping. While this software works favorably with the input data, develops the process properly and obtains partially optimized results, it is not inherently designed to be used in the game industry, preventing it from being an all-inclusive application which can be used independently.

Finally, Houdini is essentially a procedural game and media-oriented tool that comprises a wide range of toolsets and covers different criteria of 3D graphics. Due to its highly efficient performance, it can be employed independently in a game environment production, especially when the artist intends to develop the entire project based on imagination instead of using an existent geographical data. This software supports limited features and file formats for importing GIS data which makes it imperfect when the environment is supposed to be built on an imported area topology.

The range of file formats supported by an application indicates how capable that application is to read or write different types of datasets, also to exchange data with other applications. In other words, one application’s ability to read more file formats means that there is a larger number of relevant tools to provide this application with input data, while its ability to write a wider range of file formats as output data, makes it more valuable to use in various projects’ workflow. Table 2 depicts a list of file formats, supported by the reviewed applications.

Table 2 List of file formats supported by the applications

File Support for shape, Graph and static model import	Ghost Town	CityEngine		Houdini	
		Read	Write	Read	Write
ABC (Autodesk)			√		
Alembic (Sidefx Houdini data)			√	√	
.BGEO				√	√
.BPOLY				√	√
COLLADA DAE		√	√		
.D				√	√
.DXF		√		√	√
.EPS				√	
.FBX (Autodesk)		√	√		
.FGDB		√			
.GEO				√	√
KMZ/KML (ArcGIS Earth data)		√	√		
.LW				√	
.MAX	√				
.MED				√	
.OBJ (Autodesk)	√	√	√	√	√
.OSM	√	√			
.PC				√	√
.PLY				√	√
.POLY				√	√
.RIB					√
.SDL				√	
.SHP		√		√	
.STL				√	√
.TIF (Terrain and Texture import)		√			
.VOB (e-on Vue)			√		
.XML	√				

Each program works with a set of file formats to import or export data. GhostTown as plugin for 3Ds Max, communicates through four file formats supported by its host software. CityEngin imports data with 9 file formats for graph, shape, static model, terrain and texture while it can export data to different applications such as Autodesk Max and Maya, houdini etc. Finally, Houdini reads about 16 data types and exports data via about 11 file formats.

5 CONCLUSION

The objective of this thesis is to define an optimal list of features to consider in the 3D modelling tools for facilitating quick and efficient process and achieving high-quality outcomes. The fundamentals of video games environment design were studied, and feasibility and advantages of relevant methods were explored. GhostTown 1.0 plugin for 3Ds Max were examined through creating several types of environment elements. Then, two vastly used procedural 3D generating applications, namely Esri CityEngine and Houdini were technically inspected, and their capabilities and respective feature sets were reviewed. As a result, a final feature list was compiled providing information on essential capabilities of applications in handling a series of consecutive phases in procedural urban modelling including geospatial data entry, road and street networks, parceling, building constructions, Level-Of-Details, texture mapping and finally optimization of the result to export to the game engine. Then a table was generated based on the feature list which benchmarked GhostTown, CityEngine and Houdini.

Although none of these applications covered all the listed feature in the table perfectly Ghost town was not recommended due to its flaws for optimizing the assets' geometry and texture mapping. As an independent application, Houdini FX (version 15 and later) collaborates with video game modelling process most efficiently. However, in case the game environment is intended to be constructed on an existing map, it may be appropriate (or even the best) to combine the use of Houdini and CityEngine. Though Houdini can import a wide range of file formats, considering the field of supported geometry types in Houdini and CityEngine, the CityEngine can communicate with GIS data more adequately than Houdini. Houdini's strong toolset for creating optimized 3D assets and CityEngine's solid capability for engaging with GIS data could largely strengthen the urban modelling process on both ends.

There may of course be other scenarios that require game developer teams to determine a flexible use of these applications for wide range of purposes. Selecting a software for city modelling depends greatly on the project's expectations, volume of details, preferences for trade-off between cost and aesthetic, efficiency restrictions, priority of the regarded features etc. This study encourages 3D artists to meticulously plan their own strategy by matching the features listed by this thesis to their own project

demands. This will ensure that the artists will benefit from all the favorable aspects of such feature-rich computer applications.

Limitations of this study and opportunities for future research

Although the current thesis was carefully prepared and reached its defined objectives, it presents limitations and shortcomings which can be considered and addressed by the future research. This topic has the potential to be enriched by benefiting from the opinions of the experts in the field and through deeper investigation of existent scientific articles. Moreover, including a greater variety of procedural modeling applications in the study, comparing and benchmarking their constituent features can deliver more interesting, deliberate and informative results.

6 REFERENCES

Dargie, James. (2007). Modelling techniques: Movies vs. Games. Computer Graphics. 41.. 10.1145/1268941.1268944.

Desktop.arcgis.com. (2017). What is Esri CityEngine | ArcGIS Desktop. [online] Available at: http://desktop.arcgis.com/en/cityengine/latest/get-started/overview-cityengine.htm [Accessed 7 Oct. 2017].

Desktop.arcgis.com. (2017). Tutorial 7: Facade modeling—CityEngine Tutorials | ArcGIS Desktop. [online] Available at: http://desktop.arcgis.com/en/cityengine/latest/tutorials/tutorial-7-facade-modeling.htm [Accessed 16 Oct. 2017].

Esri (2017). Workflow to create 3D city from GIS data. [image] Available at: http://desktop.arcgis.com/en/cityengine/latest/get-started/GUID-6A1A6FCA-3637-4E10-BF8A-1F137A84792C-web.png [Accessed 7 Oct. 2017].

Fournier, Matthew John. (2014). Procedural City Generation and Techniques for Game Designers. Master of Art in Visual Effects. The Savannah College of Art and Design.

Hanan, J. (1992). Parametric l-systems and their application to the modelling and visualization of plants. Regina, Saskatchewan: Faculty of Graduate Studies, University of Regina.

Kelly, George and McCabe, Hugh (2006) "A Survey of Procedural Techniques for City Generation," The ITB Journal: Vol. 7: Iss. 2, Article 5.

Lifewire. (2017). 7 Common Modeling Techniques for Film and Games. [online] Available at: https://www.lifewire.com/common-modeling-techniques-for-film-1953 [Accessed 1 Oct. 2017].

Lifewire. (2017). So, What Is Texture Mapping? Here's Your Surfacing 101 Crash Course. [online] Available at: https://www.lifewire.com/texture-mapping-1956 [Accessed 2 Oct. 2017].

Mullen, T (2009). Mastering Blender. 1st ed. Indianapolis, Indiana: Wiley Publishing, Inc. ISBN 9780470496848

Pcmag.com, (2004). Procedural Texture. [online] Available at: https://www.pcmag.com/encyclopedia/term/49743/procedural-texture

QGIS, GIS4Design (2017). QGIS-HOUDINI WORKFLOW 1. [image] Available at: https://gis4design.files.wordpress.com/2016/01/qgis-houdini-wf1-flowchart1.jpg?w=660 [Accessed 14 Oct. 2017]

QGIS.org. (2017). Welcome to the QGIS project! [online] Available at: http://www.qgis.org/ [Accessed 13 Oct. 2017].

Rfx.com. (2017). RFX Inc. [online] Available at: http://rfx.com/products/169 [Accessed 9 Oct. 2017].

RFX Inc. (2017). Dynamic city blocks and street curves. [image] Available at: http://www.esri.com/software/cityengine/~/media/Images/Content/Software/cityengine/graphics/f3-thumb2.jpg [Accessed 14 Oct. 2017].

RFX Inc. (2017). Map-Controlled City Modeling. [image] Available at: http://www.esri.com/software/cityengine/~/media/Images/Content/Software/cityengine/graphics/f8-thumb2.jpg [Accessed 14 Oct. 2017].

RFX Inc. (2017). Facade Wizard. [image] Available at: http://www.esri.com/software/cityengine/~/media/Images/Content/Software/cityengine/graphics/f6-thumb2.jpg [Accessed 14 Oct. 2017].

Sidefx.com. (2017). SideFX Releases Houdini 15 | SideFX. [online] Available at: https://www.sidefx.com/community/sidefx-releases-houdini-15/ [Accessed 12 Oct. 2017].

Sidefx.com. (2017). Poly Expand 2D. [online] Available at: http://www.sidefx.com/docs/houdini/nodes/sop/polyexpand2d#overview [Accessed 14 Oct. 2017].

William Ehrendreich, (2016). shared in Kilad website. [image] Available at: http://www.kilad.net/GTForum/viewtopic.php?f=35&t=1205

Www.autodesk.com. (2014). Introduction to UV mapping. [online] Available at: https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2015/ENU/Maya/files/UV-mapping-overview-Introduction-to-UV-mapping-htm.html

www.rfx.com. (2017). Generating cityscape based on Geospacial Data. [image] Available at: http://rfx.com/products/169 [Accessed 9 Oct. 2017].