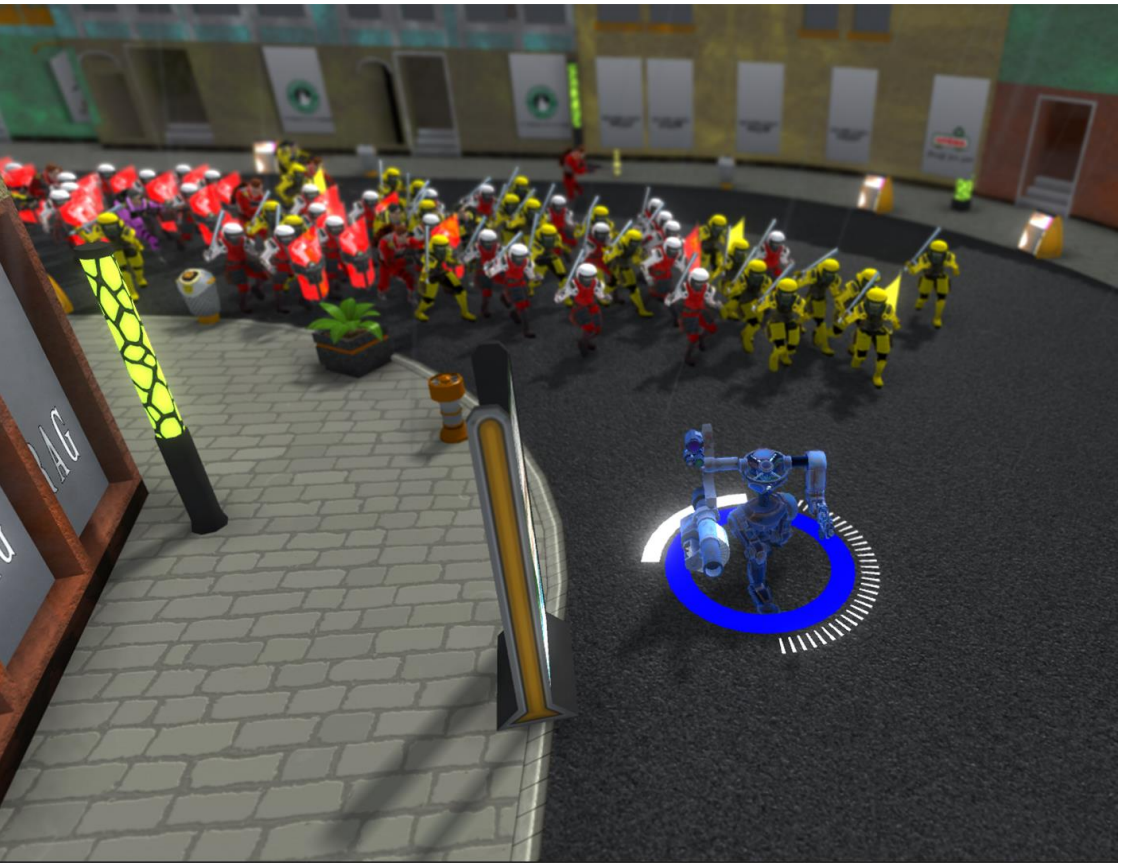
Tommi Helin

Defining Environment Art Style in a Procedural Game World



Bachelor of Engineering

Spring 2016

KAJAANIN
AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

ABSTRACT

Author: Helin Tommi

Title of the Publication: Defining Environment Art Style in a Procedural Game World

Degree Title: Bachelor of Engineering, Information Technology

Keywords: Video Game art, Art Design, Game Environment, Procedural

This thesis is about defining visual style for a procedurally generated game world and different methods in achieving a well-planned, unified art style. The thesis deals with the possibilities and issues in computer generated environment and edge cases that should be thought of beforehand.

Building an appealing procedural world is looked upon through deciding which shapes to use, finding and applying a stunning color scheme, how to handle lighting in generated world, and using shader programs to enhance the environment furthermore.

Initial purpose was to design and carry out an effective and flexible procedure to create stunning and efficient visuals and apply it for the game Spareware, which is made for Xbox One by Rust0 Games which is included as case study.

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

TIIVISTELMÄ

Tekijä: Helin Tommi

Työn nimi: Visuaalisen ilmeen määrittäminen proseduraalisessa pelimaailmassa

Tutkintonimike: Insinööri (AMK), tietotekniikka

Asiasanat: Videopelit, peliympäristö, proseduraalinen pelimaailma, visuaalinen ilme, peligrafiikka

Opinnäytetyön aiheena oli luoda toimiva menetelmä proesduraalisen peliympäristön visuaalisen ilmeen toteuttamiselle. Tavoitteena oli esitellä ja pohtia proseduraalisesti luodun peliympäristön aiheuttamia rajoitteita ja mahdollisuuksia peliympäristön visuaalisen ilmeen määrittelyssä niin tekniseltä kuin visuaaliselta kannalta.

Työssä käsitellään peliympäristön muotojen hahmottamista, visuaalisesti miellyttävän väriteeman luomista, valaistuksen hyödyntämistä ja soveltamista tietokoneen luomassa pelimaailmassa sekä pelimaailman kohentamista entisestään varjostinohjelmien avulla.

Työ toteutettiin Rust0 Games Oy:lle, jossa sitä sovelletaan Spareware–videopelin tuotannossa. Lopputuotteena oli kehysympäristö hyvän visuaalisen ilmeen toteuttamiseksi. Työ on esitelty tapausesimerkkinä opinnäytetyön lopussa.

Forewords

I would like to extend my gratitude for every living soul support me during my studies and this thesis process. Special mention to every professional game developers sharing their views and experience, my colleagues at Rusto Games, my thesis supervisor and my wife for the gratuitous assistance and support you provided.

TABLE OF CONTENTS

LIST OF TERMS

3D Model	A digital three dimensional projection of a game object on a two dimensional surface
Albedo	Color reflection of a surface
Color Scheme	A preselected set of colors
Color Wheel	An illustrative organization of colors around a circle that is designed to show relationships between primary colors
Environment Art	Visual graphics that simulate game world environment
Procedural Content	Computer generated content based on mathematical algorithms
Prop	An environmental object in the game level
Rendering Pipeline	Sequence of steps used to generate a 2D representation of a 3D scene on a display device
Shader Program	A program written to control target platform's graphics processing unit
Target Platform	The gaming platform that the game is published on

1 INTRODUCTION

The purpose of environment art in video games is to create a believable and immersive experience of a virtual game world for the player [1].

Subsiding or supplementing traditional game level design and game art with procedurally generated content has long been part of the video game industry. A lot of old computer games relied heavily on procedural content, where it was used as a way to save memory and disk space. Computer generated game levels enable game developers to add more variation for a game with less effort compared to using man hours to achieve similar results. Nowadays more and more game studios are adapting procedural content for their games due to rising costs of content creation [2, 3]. Creating visuals in a procedurally generated game world can however prove to be a bit difficult, since a lot of control is given away to gain variation [4].

Going procedural does not mean going for random. Procedural generation is about creating the guidelines for a computer to work with. Instead of being random it is based on computer algorithms. Therefore, if you give a procedural generator a same set of parameters, it will always create similar content [4]. A lot of the art design in procedural games is more about creating a set of rules than making art in a traditional sense [5]. The visual outcome is based on controlling these rules instead of doing the game assets manually as in traditional game level design.

2 DESIGN PROCESS

Starting the art design process for a procedural game should include a group of artists, programmers and other relevant members of the team. Clear and present communication between programmers and artists is imperative to achieve good outcome on any game project, but increasingly so when dealing with procedural systems [4]. This is a beneficial symbiosis for both departments where professionals of math and algorithms and professionals of visual excellence can pool their effort to create something spectacular.

Procedural game environment generation can be a done with a generator program that defines pretty much everything of how the environment looks from geometry and lighting to the color and texture of the objects. A procedural generator can also be a hybrid system consisting of handmade assets which are then placed by computer based on algorithms.

Defining the workflow early on is important. It is a good practice to discuss and choose the overall visual style at early stages, even though the visual style will live on through the development cycle of the game. Decisions about what should be done procedurally and what should be done by hand should be included in the early design process.

Commonly used ways to speed up the design process include concept art, mockup pictures or game scenes and rapid prototyping. While making a game with procedural content these can then be used as a help when deciding the process for creating the content. Concept art is a good way to communicate the overall visual style of the game [6]. This can be anything from pencil or charcoal sketches to digital paintings or 3D modeled scenes. Mockup images are made to imitate or foretell how the game could look like beforehand. Typical ways of doing a game mockup is creating a digital painting of a single game scene, or building quickly made three dimensional scenes or models to present the visual style for the entire team.

Considering procedural systems, it is a good practice to create rapid prototypes of the level generator with a programmer tasked for creating the actual logic behind the generator subprogram [7]. Creating early prototypes of the level with gray or white models lit with a simplistic lighting model is a good way for finding out interesting shapes and silhouettes that work with the environment.

Once the general guidelines for the art style are sorted out, the details should be well thought beforehand too. Rules and guides should be defined for how the actual game environment assets should be produced. At this stage it should be considered if the assets are to be created via a procedural algorithm, by hand, or as a hybrid system.

After the initial procedures for creating the art for the game are sorted out it is a good practice to write a thorough art guide document [8]. The art guide should include the general characteristics of the major guidelines for creating game assets. It should clearly present what is required of an artist to create successful and unified game art. Among things that the document should include are for example the overall feel of the game art, silhouette guide lines, color guidelines, material properties, texture maps used and any additional shader properties that should be taken into consideration. A well written environment art guide defines the environment art properties with well written guidelines filled in with descriptive pictures, such as color wheels, color guidelines, images of example 3D models, textures and so forth.

3 FORM AND SHAPE

Planning ahead the form and shape of game assets comes increasingly important when dealing with procedural game environments. A wrong rotation or uncanny shape can easily break the immersion of a believable game world. It is often thought that a credible game world follows the rules of the real world and generalizes them to be easily understandable for the player in a split of a second. However, a certain amount of disorientation and chaos is needed to make the world feel believable [9].

3.1 Defining shapes

The shapes and forms should be easily recognizable by silhouette alone. Shapes are increasingly important for people with limited color vision. Keeping the shapes of interactable or collidable objects somewhat similar and recognizable helps the player to identify the purpose of the game environment.

Variety in shapes is essential in keeping the game world interesting. An environment filled with only single type of shapes quickly becomes monotonous and boring. Using different polygonal forms and curved forms in different, yet logical applications will add greater value to the game environment. Using silhouettes in the design process helps creating strong, memorable levels [10]. Starting out with a strong abstract shapes, and adding medium sized and small details creates a better visual experience. For example a silhouette of a windmill can be recognizable, but suffer from a lack of memorable feeling. If some detail is added, it can make the same windmill look a lot more of the archetype of a windmill. (Figure 1). These ground rules should be used in every game object to make the environment feel livelier.

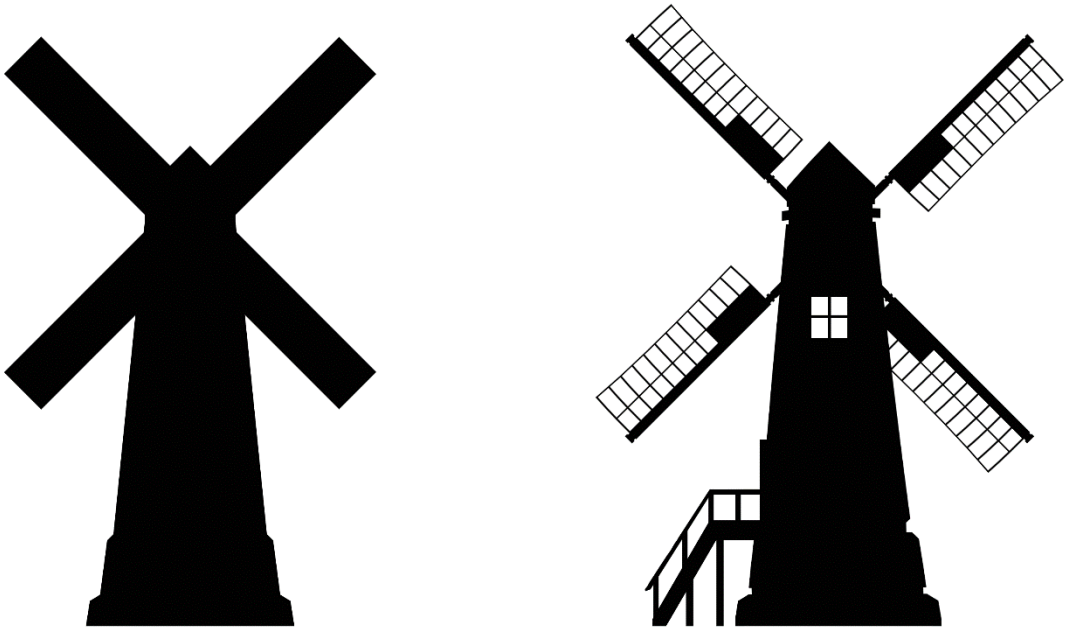


Figure 1. Side by side comparison of a dull and more interesting silhouette of a windmill.

While defining shapes it is relevant to understand and embrace modern and classical standpoints in aesthetics of shapes. Circular and curved shapes are often viewed as friendly shapes, whereas rectangular shapes are considered to be firm and stable and triangular pointy shapes are thought of as aggressive or painful surfaces [11]. Same principles can be generalized for the division lines that separate shapes from others. Curvy shapes are looked at as soft, pleasant surfaces.

Contrast of game characters' shape should also be considered against the environment. It should be decided if the character is to create a feeling of harmony or dissonance against the background. While harmonic approach will make the player feel at ease, dissonant background will make the character stand out more. These can be used to change the mood of the environment accordingly, making the player feel safe or endangered at appropriate times.

3.2 Controlling asset placement

A lot of criticism towards procedural level generation is a consequence of poor computer defined asset placement [4]. For a multitude of reasons critics tend to blame a whole branch of workflow instead of bad algorithms actually responsible for the misplaced assets.

Considering asset placement, artists tend to love to stay in control [4]. Minor obvious tasks for environment artists, such as cracks of asphalts cannot be necessarily hand placed when working with a procedural system. However, given the right parameters such placement could be just as reliable as hand placing them, and in cases, even more reliable. The generator program can be taught to recognize shapes and rotation of objects, and apply this to preset parameters of how such objects should be placed to the game world [4].

Where handiwork is still ahead of procedural systems is the handling of exceptions. Even with the best systems artists are still needed to supervise the algorithms work and fill in the parameters, at least for now.

Just as it is possible to create completely new geometry with right algorithms, procedural level generator can be taught to place additional props according to the newly created geometry. A typical example of this is a procedurally created road and evenly spaced streetlights that are aligned to the road.

While designing the assets it is crucial to keep in mind what kind of edge cases the asset might be put into (Figure 2). A slope on the surface, or a corner in a wall

will require attention while creating the asset. If the asset is placed on slope, it should have a long enough base so it will not just float in the air.

Figure 2. A side by side comparison of good and bad asset placement. Screen capture from Spareware, Rusto

To achieve a sense of realism a little offset in both object placement and rotation could be added. Procedural algorithms are often programmed to align objects along lines, such as walls or roads. Real world objects placed by humans are rarely aligned perfectly [3]. Thus perfectly aligned objects make a very unnatural or clinical feeling to the environment (Figure 3). It is still needed to remember the possible edge cases the object might be face. Trying to achieve credibility with rotational offset easily turns into an immersion breaking mistake if the object clips with the ground or other objects.

Figure 3. Side by side comparison of two lines of cars with and without position offset. Screen capture from Spareware, Rusto

4 APPLYING COLOR

A lot of things have to be taken into account while defining colors for a game level. These include but are not limited to different types of display devices, differences in peoples color sensitivity and the general feeling colors have on players.

Applying a clear color scheme in environment art is an easy way of dealing with the unification of game assets. Even though the forms and shapes of objects would be totally different, player will be able to distinguish objects to belong to a specified category. However color scheme alone cannot be trusted as the sole distinguishable factor as a good amount of people have a limited sense of color. Therefore brightness of color in addition to hue and saturation plays a key role in color design.

4.1 Foreground & Background

Foreground in this context means the part of the environment that is interactable by the player, whereas the background refers to the parts that is not. Foreground includes the level props, areas where the player can move and objects that the player character can interact with. Any objects the player can interact with should have a point of focus [12], a decisive difference from everything else on the ground. Point of focus can be either color or done with geometry. If point of focus is done with a color, it can be pointed out via choosing strong contrast, strong saturation or complementary color against the surrounding environment.

Difference in color of the foreground and background should be easily perceptible, preferably by strong difference in contrast and saturation. This can be divided into thinking the game scene as a layer. One layer has the interactable objects and props, while the other consists of background, or canvas. While objects in object layer can be in contrast or complement each other, they all should differ from the background [13].

Game props should be easily identifiable from the player or artificial intelligence controlled characters. Therefore it would be a good idea to have the colors of environment props differentiate from character models in either hue, saturation or brightness of colors.

People are subject to identify objects in spaces. Common human behavior is to look at small fraction of different color over large area than other way around [13]. Hence, attention color should be encouraged to use rather in small portions, as opposed to huge amounts.

4.2 Selecting a color scheme

A good understanding of color theory helps in creating visually appealing game levels. Selecting a color scheme based on known and accepted models could prove to be a good idea, since they are proven to be visually engaging.

Over the years, people have begun to see different meanings in colors. People associate certain colors with certain feelings, such as blue meaning harmony and red indicating danger. A lot of color associations are bound to cultural factors and they even differ depending on the context. For example red color is looked at as a symbol of beauty in Russia while it indicates danger in vast majority of Western European cultures. Although these associations are not definite, they should be taken into account while designing the color scheme.

Using multiple colors together also carries out different feelings depending on colors used. Using complementary colors, the colors opposite to each other on color wheel creates more feeling of disharmony, whereas using analogous colors that are adjacent to each other are more easily associated with harmony. Furthermore, color scheme should match the overall theme of the game environment. Colors in game levels should embrace the feelings that the game developers want to give to the player.

To aid in the process of selecting colors, a set of useful tools can be applied varying from color scheme cards to interactive color wheels. A wide range of predefined color schemes also exist to speed up the process of selecting colors. Furthermore, a lot of game companies have shared their vision on how to build effective game environments. There even exist formulas for selecting interesting color schemes [14].

Figure 4. Screen capture of interactive color selector Adobe Kuler [15]

A procedural approach to applying color can also save time. It is quite possible to create a ruleset that produces visually appealing secondary or supplementary colors. These are then automatically applied just by selecting the main color or few colors and letting the computer do the rest of the job for you [4]. A lot of odd edge cases are possible in procedural color defining and a thorough supervision is required to assure the color scheme created by the algorithm is visually pleasant because the program does not know the idea of a pleasant color. The generator program can however be taught to recognize acceptable cases, but human eye is still more refined to understand the concept of color theory. The supervision process can be further advanced by creating screen capture algorithms or easy to use test environments to assure the algorithm is working properly.

An even more refined approach is utilizing a shader program with color masks. Color masks are images that are combined to a texture file with a desired effect. With careful planning color masks can be packaged into a single file that has multiple individual masks using the red, green, blue and alpha channels of a digital image [12]. Even more masks can be used with a careful planning of UV-maps, the coordinates used to define which part of texture map is used on which part of the 3D model [12].

5 DEPTH IN LIGHTING AND SHADING

Selecting the right lighting engine is an important task to guarantee the quality of the end result in procedural game worlds. The feeling created by correct lighting is hard to replace by other means. However, should the selected lighting model have considerable amount of technical limitations, a lot can also be faked.

Lighting in forward rendering is calculated per object, which limits its amounts of visible lights. [16] Forward rendering pipeline can sometimes prove to be the right option despite its lack in real time lights per visible area. Due to deferred rendering pipeline occupying more of the graphics processing unit's capacity with later passes, more can be done in Geometry buffer with forward rendering.

Deferred shading is often considered an industrial standard and is in many cases selected without a second thought. Deferred rendering renders the game environment as a set of two dimensional images rather than each object individually. [16] Lighting is then calculated comparing light source positions to these images. Compared to forward rendering deferred rendering gives a serious edge in the quality of lighting and number of real time lights per visible area.

5.1 Shaders in game environment

Utilizing shader programs in the game environment gives a feeling of depth often not achievable by textures alone. Shading can also greatly reduce the time needed for creating visually appealing output for the game.

A variety of post processing effects can be applied to make the game feel even more refined. A typical way to enhance the game environment is to use prebaked light maps. This is done with texture maps that are combined to textures of the game objects. Prebaked light and reflection maps are often excluded in procedural game environments, due to the fact that the game level is usually generated runtime. Screen space effects are an efficient way of adding a finishing touch to the game if prebaked light maps are not a possibility.

Screen space effects are calculated only per visible area [17], thus saving processing time compared to prebaked effects. This can be applied to make the game world's materials feel more plausible via real time light mapping, reflections and ambient occlusion effects.

Although not as accurate as prebaked effects, screen space effects can create plausible enough visuals that still bring a great additional value to the game environment. Since screen space effects are post-process effects, being rendered after the main rendering pipeline, using them can also serve as workload distribution for the target platform's graphics processing unit.

A good reflection shader brings a strong feel of depth and feeling of authenticity to the materials. Real time reflections can however be expensive on processing budget and should be thought of carefully before implementing into the game. If within processing capacity, reflections bring a feel of material hard to replace otherwise. Fortunately a pre-baked cube map reflection can be used if no real time reflection is possible. Cube map is a set of six images representing cubical area around the camera. Visible part of this cube map is reflected back to the camera based on the reflective surface and the camera angle.

Using a solid light enhancement shader is a good way to make lighting feel less flat. A well programmed bloom shader brings a rich addition to the lighting in environment, adding glare over the lightest areas of the game [18]. Bloom should still be used with consideration. It is easy to go overboard with the bloom shader and a badly done bloom shader is misleading for the player and is subject to criticism [19].

Figure 5. Screen capture of Spareware with no post processing image-effects and reflections.

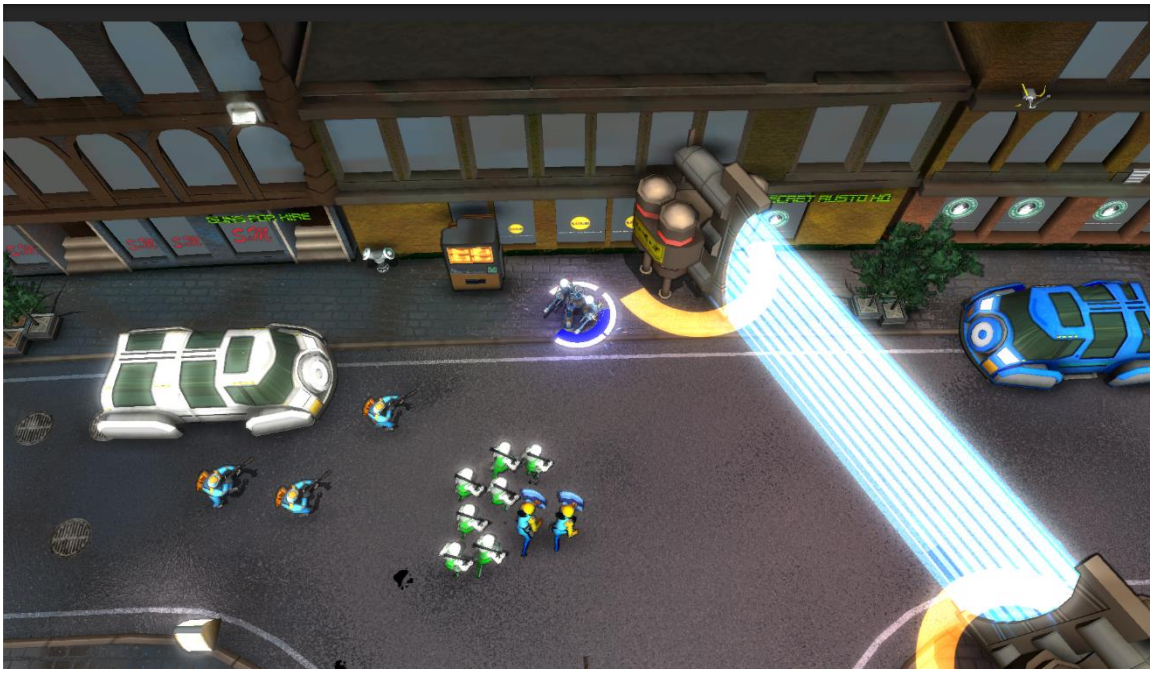


Figure 6. Screen capture of Spareware with post processing image-effects and cube map reflections on.

5.2 Lighting the environment

Choosing the lighting, light color and giving the parameters of light placement is essential in creating a believable environment.[20] Lighting is a key factor in creating the mood of the game environment, since people's feelings are heavily affected by real world lighting as well. Different lightings carry different moods and feelings. Selecting the light source and having lighting come from the source is a necessary step towards credibility. Floating lights coming out of nowhere are prone to make environment look dull and unconvincing.

A lot can be done with a few lights and a simulated sunlight alone. Adding spot or area lights will enhance the liveliness of the environment. It is however easy to go overboard with lighting [20]. Adding too many lights to the game is often as unnatural as too few. Selecting a known lighting model miming well known lighting models is a good way to play it safe since a lot of lighting algorithms already try to impersonate models used by classic painters.

Indirect light is light reflecting from one surface to another, and is helpful in making the environment feel plausible. This is not just for game environments aiming to be as realistic as possible, but also for creating stylized effects in lighting. Calculating real time indirect lighting is an expensive process resource wise and often has to be circumvented.

Ambient lighting is not to be overlooked. Ambient light can be thought of as a fill light that does not derive directly from any light source but is added to every object in the game scene by default. Ambient lighting can create a credible enough feeling of indirect lighting [21]. If generating a physically accurate indirect lighting is not possible, a simple single colored ambient light softening the shadowy side of geometry can build a huge leap in game environment's credibility. This can be further enhanced by modifying the ambient lighting with a gradient lighting along upwards world axis.

Lighting in a procedural world can be a bit tricky. A typical way to save rendering time is to bake light maps by baking the lights to textures [22]. This is a time consuming and performance wise expensive procedure that is ruled straight out in

many cases. This of course depends on the way the levels are generated. If the environment is loaded from bigger chunks with fixed light sources, lights can be baked beforehand for each area. Lighting can be baked during the game runtime, in the loading phases of the game, but might extend the loading times to unbearable amounts.

6 SPAREWARE CASE STUDY

Spareware is a twin-stick top-down shooter game for Xbox One. Spareware is made with Unity game engine. The setting is in dystopian future Helsinki where the ever present direct democracy of vote is oppressing the free will of people.

The initial task was to do a set of guidelines for environmental art style reboot to fit the premade futuristic robots. The environment style should fit the idea of a fake utopia, and be easily distinguishable from player and enemy characters.

6.1 Designing the environment art

Testing was done for decrepit, cartoony and a highly modernized game worlds. The environment art style for Spareware was chosen to be a combination of each, with the emphasis on strong and bright, slightly cartoonish colors with a hint of tainted surfaces. Background was chosen to be simplistic road textures on road and street pavement.

A general color scheme was selected to be orange. Orange was the governing color in just about anything ranging from UI to game logos and game environment assets. Environment color scheme was chosen to go with the overall color scheme. Therefore every smaller, collidable object was to have a portion of slightly reddish orange color, or its composite color, teal to distinguish it from the rest of the environment.

Environmental art guides consisted of an overall general art style document and a more specific art guide for each biome, so all artists were on the same page considering the art style.

Overall art guide included the guidelines for each part of game art including environment art subdivided to smaller partitions, such as characters, environment assets and props. The overall art guideline described generalized color schemes, shaders used, material properties and a general guideline on what kind of setting the asset should fit.

The smaller but greater in number biome specific guidelines included more specific material properties with more detailed values, color scheme in greater detail and a guideline on what sort of feeling was to be gone after. The documents also included reference images and videos to further enhance the unified vision of the entire art team.

Material properties were added to the design documents. Material properties in environment props consisted of two texture maps. The albedo map is used for the object's color properties and the emissive light color map is used to simulate light emitting from certain parts of the object (Figure 7). Textures were done using texture atlases shared by multiple 3D models to conserve space and memory. Additional properties included a light map for lighting color, and a cube map for reflective surfaces such as metal and glass. Although a satisfactory result was gained, a lot more variation could have been achieved by employing additional color maps and changing colors or adding decals or wearing procedurally.

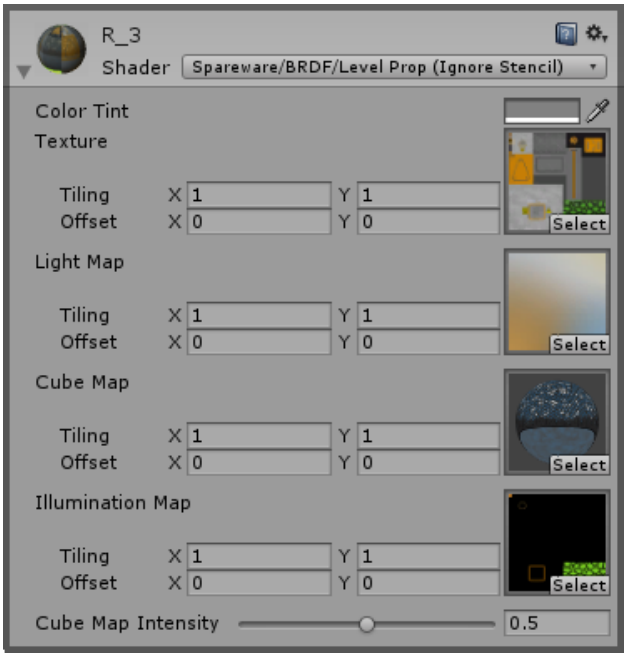


Figure 7. Screen capture of environment asset's material properties.

6.2 Level Generation

The level generation system was chosen to be a hybrid system, consisting of computer generated streets and buildings, supplemented with hand modelled environment props, placed with an algorithm. The placing algorithm was able to recognize various geometrical areas in the environment, such as wall and street directions and populate the streets and sidewalks with correctly placed props. Population logic could have been a bit livelier and more offset and variation could have been added to gain little more in the plausibility in the game world.

The level generator consisted of different biomes that represent different eras in architecture. Whereas most biomes were made to represent older style of architectural Helsinki, there were some a bit more decayed and even more futuristic, commercial type of districts. Each biome had varying textures on buildings and streets. Color scheme for the buildings was cartoony pastel colors, based on real world Helsinki Art Nouveau architectural style. The colors differ from brown pastel colors of suburban environments to the bright and saturated colors of downtown Helsinki.

6.3 Fine tuning Spareware

Environmental effects including rain and cube map reflections were added to make the game feel livelier. Particle effects were used to generate more depth to the game world in the form of smoke and electric sparks.

Tryouts with deferred lighting and multiple lights were done. While additional point lights gained in deferred rendering created a more beautiful atmosphere, deferred rendering pipeline occupied too much space from graphic processing unit’s stencil buffer. The stencil buffer was needed for other purposes, therefore customized forward rendering was selected as Spareware’s lighting model.

The lighting model was eventually decided to be custom written bidirectional reflectance distribution function style lighting model based on deferred rendering pipeline. Bi-directional Reflectance Distribution Function based lighting is a local

illumination model that is based on calculations on how much light is reflected in particular direction after light has reflected off the surface of the 3D model [23]. Purpose for selecting a customized lighting model was to utilize more lights per visible area, while using forward rendering pipeline and gain more control over the lighting of an object. The mood in lighting was chosen to be cartoonish, with a possibility of rain which changed the game to be a bit moodier every once in a while.

A set of post processing effects was used to bring Spareware the look it has. A middleware screen space bloom effect was implemented to bring additional shine for the bright parts of the game. This added a feel of material that would have been hard to do without

A few different approaches for real time reflections were tried out. Even though the tested real time reflections gave a lot to the overall feeling and visual appeal all tried approaches proved to be too costly for target platform's rendering pipeline and could not be justified. Therefore a pre-rendered cube map reflection was selected to give a feeling of reflective materials with lot less processing cost.

Being a procedural game world generated at runtime, a prebaked ambient occlusion map was out of the question. To bypass this, most assets have ambient occlusion prebaked into the albedo texture maps. To emphasize the ambient shadows furthermore, an optimized screen space ambient occlusion effect was made to make the lighting look more plausible, enhancing the occlusion shadows forming between edges of two object or indents.

An edge detection effect was used to bring a slight cartoony outline for the objects to further distinguish objects from each other. Edge detection was used moderately as a semi-transparent effect around the game objects. This was done instead of fully cartoony black outlines often seen in similar games because opaque outlining would have gotten too much of the player's focus.

Figure 8. Screen capture of the final game Spareware, Rusto

7 CONCLUSION

Although challenging, procedural game worlds bring huge possibilities in creating variety to the game worlds with relatively infinite possibilities. Traditional digital art creation tools have already adapted a huge variety of procedural elements from procedurally generated character skeletons used in animation to copying and modifying 3D models rapidly. Because of its huge possibilities procedural art creation is by no means a subsiding trend.

The day when procedural algorithms are able to generate fully viable game environments mostly by themselves may still be a good way in the future. Even if procedural algorithms would someday be able to replace entire art teams in the game development industry, a solid art design is still needed to lay the grounds for the algorithms to work on.

Despite a lot of critic and controversy towards procedural game level creation, procedural systems gain more ground as the content creation algorithms are getting ever better by the day. Procedural generation can be a valuable extension, and in some cases even a replacement for the art teams, allowing game studios to create more content with less manpower.

REFERENCES

[1] Castillo T, Novak J. Game Development Essentials: Game Level Design. 1st ed. Clifton Park, NY, USA: Delmar Learning; 2008. p. 147.

[2] Young S. The Future Is Procedural 2009; Available at: http://www.escapist-magazine.com/articles/view/video-games/columns/experienced-points/6418-The-Future-is-Procedural. Accessed 1.6.2015.

[3] Favison K. Sponsored: Go Procedural – A Better Way to Make Better Games 2015; Available at: http://www.gamasutra.com/view/news/233899/Sponsored_Go_Procedural__A_Better_Way_to_Make_Better_Games.php. Accessed 12.1.2016.

[4] Duncan G. Art Direction Bootcamp : How I Learned to Love Procedural Art 2015; Available at: http://www.gdcvault.com/play/1021805/Art-Direction-Bootcamp-How-I. Accessed 15.12.2015.

[5] Epic Games Inc. Procedural Buildings; Available at: http://www.gdcvault.com/play/1021805/Art-Direction-Bootcamp-How-I. Accessed 12.1.2016.

[6] Raymond J. Concept Art: What Is Concept Art and Why Is It Important? 2014; Available at: http://artistryingames.com/concept-art-concept-art-important/. Accessed 9.6.2016

[7] Tulleken H. Rapid Game Prototyping: Tips for Programmers 2014; Available at: http://www.gamasutra.com/blogs/HermanTulleken/20140119/208901/Rapid_Game_Prototyping_Tips_for_Programmers.php. Accessed 7.6.2016

[8] Telang K. Game Art Bible - Secret Sauce to Making Great Game Art 2013; Available at: http://www.slideshare.net/pencillati/game-art-bible-secret-sauce-to-making-great-game-art/7. Accessed 9.6.2016

[9] Chmielarz A. The Secret of Immersive Game Worlds 2014; Available at: http://www.theastronauts.com/2014/03/secret-immersive-game-worlds/. Accessed 12.1.2016.

[10] Galuzin A. Silhouette Design Game Environments 2010; Available at: http://www.worldofleveldesign.com/categories/game_environments_design/silhouette-design-game-environments.php. Accessed 30.4.2016.

[11] Solarski C. Aesthetics Of Game Art And Game Design 2013; Available at: http://www.gamasutra.com/view/feature/185676/the_aesthetics_of_game_art_and_.php. Accessed 8.5.2016.

[12] Bronwen G. Shading a Bigger, Better Sequel 2010; Available at: http://www.valvesoftware.com/publications/2010/GDC10_ShaderTechniquesL4D2.pdf. Accessed 13.5.2016.

[13] Anhut A. Color Theory For Game Design 1 of 4 - Fundamentals 2014; Available at: http://howtonotsuckatgamedesign.com/2014/11/color-theory-game-design-1-fundamentals/. Accessed 8.5.2016.

[14] Seitz T. Picking a Color Palette for Your Game's Artwork 2012; Available at: http://gamedevelopment.tutsplus.com/articles/picking-a-color-palette-for-your-games-artwork--gamedev-1174. Accessed 8.5.2016.

[15] Adobe Inc. Adobe Kuler; Available at: https://color.adobe.com/. Accessed 15.5.2016.

[16] Owens B. Forward Rendering vs. Deferred Rendering 2013; Available at: http://gamedevelopment.tutsplus.com/articles/forward-rendering-vs-deferred-rendering--gamedev-12342. Accessed 3.6.2016

[17] Wronski B. The future of screenspace reflections 2014; Available at: http://www.gamasutra.com/blogs/BartlomiejWronski/20140129/209609/The_future_of_screenspace_reflections.php. Accessed 5.12.2015

[18] Kalogirou C. How to do good bloom for HDR rendering 2006; Available at: http://kalogirou.net/2006/05/20/how-to-do-good-bloom-for-hdr-rendering/. Accessed 9.6.2016

[19] Gallant M. Bloom Disasters 2008; Available at http://gangles.ca/2008/07/18/bloom-disasters/. Accessed 9.6.2016

[20] Finch A. How to light your basic game environment 2015; Available at: http://www.creativeblog.com/video-games/how-light-your-basic-games-environment-51514875/2. Accessed 8.6.2016

[21] Unity Technologies. Unity 5 Lighting and Rendering; Available at: https://unity3d.com/learn/tutorials/topics/graphics/unity-5-lighting-and-rendering. Accessed 9.6.2016

[22] Masters M. Light Up Your World: How Lighting Makes All the Difference For Your Games 2014; Available at: http://blog.digitaltutors.com/understanding-the-importance-of-lighting-for-games/. Accessed 28.4.2016.

[23] Rusinkiewicz S. Bi-Directional Reflectance Distribution Functions 1998; Available at: http://www.cs.princeton.edu/~smr/papers/brdf_change_of_variables/brdf_change_of_variables.pdf. Accessed 3.6.2016.