Pontifícia Universidade Católica de São Paulo

Gabriel Cavalcanti Marques

Introdução ao desenvolvimento de jogos digitais utilizando o motor de jogo UDK

Mestrado em Tecnologias da Inteligência e Design Digital

São Paulo

2015

Pontifícia Universidade Católica de São Paulo

Gabriel Cavalcanti Marques

Introdução ao desenvolvimento de jogos digitais utilizando o motor de jogo UDK

Mestrado em Tecnologias da Inteligência e Design Digital

Dissertação apresentada à banca examinadora da Pontifícia Universidade Católica de São Paulo, como exigência parcial para a obtenção do título de MESTRE em Tecnologias da Inteligência e Design Digital, sob orientação do Prof. Dr. Luís Carlos Petry.

São Paulo

2015

Banca Examinadora

_____

_____

_____

Agradecimentos

Agradeço ao Criador pela jornada até aqui. A Skadi, Odin e Thor pelas horas que a fé falhou. Agradeço também ao meu pai que lutou e sacrificou tanto para que meus sonhos pudessem se tornar realidade. Agradeço a Winna Zansavio pela ajuda imprescindível para construção desse trabalho e do meu crescimento pessoal. Agradeço a Luís Carlos Petry e Arlete dos Santos Petry por me guiarem na construção desse trabalho e me prepararem para essa jornada que é a vida acadêmica. Por último e não menos importante agradeço a Suely Zansavio por ajudar sempre que pode a realizar esse sonho.

Resumo

A pesquisa aborda a questão do desenvolvimento de jogos digitais utilizando o motor de jogo UDK. A mesma visa apresentar os potenciais de desenvolvimento do motor de jogo UDK, abordando os aspectos e as especificidades de produção de um jogo digital e as etapas ontológicas de desenvolvimento como: *brainstorm*, arte de conceito, modelagem tridimensional e design de nível. Pretende-se também produzir material para ser utilizado como guia no desenvolvimento de jogos digitais com o motor de jogo UDK. Encontramos nossa justificativa no fato de que a UDK é um motor de jogo free largamente utilizado para jogos de alto padrão de qualidade, porém encontra-se carente de materiais específicos introdutórios. Teoricamente essa pesquisa fundamenta-se nos conceitos de desenvolvimento abordados por autores como Rabin (2012; 2013), Novak (2010; 2011), Thorn (2011), entre outros, adotando metodologia voltada para os aspectos artísticos ontológicos do desenvolvimento de jogos digitais, segundo Petry (2003). Voltada para a produção de um tutorial, a pesquisa também visa à criação de um protótipo navegável ilustrando como o material do tutorial pode resultar em um jogo digital ou ambiente navegável com um alto padrão de experiência estética e qualidade.

Palavras chave: Jogos Digitais; UDK; Topofilosofia; Tutorial.

Abstract

The research addresses the issue of development of digital games using UDK game engine. It is to provide the potential for development of the UDK game engine, addressing the aspects and characteristics of a digital game production and the ontological stages of development as: brainstorming, concept art, three-dimensional modeling and level design. We also intend to produce material for use as a guide in the development of digital games with the UDK game engine. We find our justification in the fact that the UDK is a free game engine widely used for high-quality standard games, but is lacking in specific introductory materials. Theoretically this research is based on the concepts of development addressed by authors such as Rabin (2012; 2013), Novak (2010; 2011), Thorn (2011), among others, adopting methodology directed to the ontological artistic aspects of the development of digital games. Theoretically, this research is based on the concepts of development addressed by authors such as Rabin (2012; 2013), Novak (2010; 2011), Thorn (2011), among others, adopting methodology directed to the ontological artistic aspects of the development of digital games, according to Petry (2003). Focused on the production of a tutorial, the research also aims to create a navigable prototype illustrating how the tutorial material may result in a digital game or inland environment with a high standard of aesthetic experience and quality.

Keywords: Digital Games; UDK; Topophilosophy ; Tutorial.

Sumário

Introdução

Nossa pesquisa se relaciona com um trajeto de vida que se inicia em 2009 no momento em que entrei na primeira turma do curso de Tecnologia em Jogos Digitais da FMU. Onde houve pela primeira vez o contato com os fundamentos teóricos do desenvolvimento de *games*, e de alguns *softwares* para a produção dos mesmos.

Utilizando o *3D Studio Max* de forma autodidata, vinculado à tentativa de desenvolvimento com o motor de jogo *Neoaxis*, passando a ter experiência da complexidade do desenvolvimento de um jogo digital tal como uma melhor noção da importância de um programador para o desenvolvimento de um jogo.

Ter um programador competente para o desenvolvimento de um jogo no Brasil e em algumas partes do mundo é uma tarefa difícil. A parte artística dos jogos digitais envolvendo tanto o desenvolvimento tridimensional como bidimensional, são mais atrativas para a maior parte dos profissionais que decidem fazer carreira nessa área.

Houve experiências com outros motores de jogo como a *Unity*, *Realm Craft*, *3D Rad* e *CryEngine*, todas ótimas ferramentas para o desenvolvimento, contudo a que oferece um fluxo de desenvolvimento mais contínuo sem esbarrar tanto nas dificuldades de programação é a *Unreal Engine 3*.

Tivemos a sorte de em 2010, a *Epic Games* detentora dos direitos e da produção da *Unreal 3* começar seu programa gratuito para desenvolvimentos, surpreendentemente uma *engine* de produção *Triple A* estava abrindo seu *software*, lógico que com algumas limitações completamente contornáveis, para a comunidade que não teria no momento de começo de carreira 35 mil dólares para ter acesso a sua ferramenta. Nascia então a UDK da Epic Games, *Unreal Development Kit*, gratuita para utilização não comercial e com um sistema favorável para quem for utilizar esta para fins lucrativos. A UDK estava indo ao encontro de um movimento no mercado que começava a fazer motores de jogo como a *Neoaxis*, provavelmente uma das percursoras do movimento de

utilização *free* de ferramentas de desenvolvimento para *games*.

A *CryEngine* no mesmo período, também iniciou seu programa gratuito, o panorama era super favorável, dois motores de jogo de produção de *games*, consolidados no mercado, estavam disponibilizando suas ferramentas para utilização, mas como dito anteriormente a programação ainda que crucial e não excluível do desenvolvimento ainda é para muitos uma grande dificuldade. Neste quesito a UDK se sobressai mediante todas as outras, com sua linguagem de programação lógica visual, o *Kismet,* da qual o usuário tem a possibilidade de fazer mil portas abrirem e fecharem sem a necessidade de revisar algumas linhas de código e fazer isso em uma *pipeline* de tempo real, acompanhando o deslocamento das portas acontecer. Essa da qual foi uma das razões pelo nosso interesse para pesquisar especificamente o desenvolvimento de um ambiente navegável/jogo nesse motor de jogo.

Por isso optamos por utilizar a UDK como ferramenta de desenvolvimento para os projetos, como todo motor de jogo, ela possui um nível de complexidade muito grande, existem várias formas de produzir um elemento e nem sempre o mais fácil deles é de fato o mais eficiente. Na UDK não é diferente, peculiaridades de desenvolvimento e pré requisitos de implementação para dentro de sua plataforma de editor precisam serem cumpridas, pré requistos para desenvolvimento de *Assets* de arte. É nesse momento que a presente pesquisa se torna relevante, oferecer a comunidade uma forma de desenvolvimento de qualidade utilizando a UDK, abrangendo parte de sua complexidade e servindo como guia introdutório para aqueles que desejam desenvolver games com a *Unreal 3*. Apresentando na forma de um ambiente navegável/jogo, a pesquisa oferecerá de forma fundamentada as etapas iniciais para produção de um *game* utilizando a UDK, tal como demonstrará a possibilidade de produção de um ambiente navegável/jogo de qualidade e padrão profissional, que proporciona experiência estética, sendo o mesmo feito em uma ferramenta de motor de jogo *free*.

Em nossa caminhada da presente pesquisa nos deparamos com a necessidade de um maior aprofundamento nesse estudo pelas necessidades advindas do projeto da Ilha dos Mortos, um projeto conjunto, sob autoria e supervisão do Prof. Dr. Luís Carlos Petry.

Para tal, Nossa pesquisa tem como objetivo geral pesquisar o motor de jogos UDK, demonstrando através do tutorial de um protótipo de um jogo digital/espaço navegável,

todas as etapas do desenvolvimento de um jogo em UDK. Para alcançar o objetivo objetivamos pontos e questões específicas, são elas:

Levantar/Investigar na bibliografia especializada primeiramente os conceitos técnicos dos jogos digitais como: que é um jogo digital, etapas de desenvolvimento de um jogo digital, como *brainstorm*, arte de conceito (*concept art*), modelagem tridimensional, o que é um modelador tridimensional, design de nível (*level design*) entre outros o que é um motor de jogo, o que são *assets* de arte e para que servem, afim de termos um escopo teórico conceitual bem definido das etapas de desenvolvimento de um jogo antes de adentrarmos especificamente no terreno do motor de jogo UDK e da produção do tutorial de um jogo/ ambiente navegável;

Assim primeiramente vamos demonstrar as etapas de desenvolvimento de um jogo digital: O *brainstorm*, a arte de conceito, o *game design*, o design de nível*;*

Pesquisar e catalogar a história do motor de jogos UDK, dedicando assim todo um capítulo da presente dissertação para tal**.**

Identificar todas as etapas básicas de produção minuciosamente, de um protótipo de jogo digital no motor de jogos UDK, a fim de demonstrar através de um tutorial a produção de um ambiente navegável de qualidade no motor de jogo UDK. Dando maior atenção para as peculiaridades específicas e para as exigências da UDK.

Pesquisar e demonstrar a viabilidade de produção de um trabalho profissional de qualidade com um motor de jogo *free*, tal como levantar jogos de qualidade *Triple A* produzidos no mesmo;

Compreender, com base no caminho percorrido, a construção de um ambiente navegável no motor de jogo UDK e a viabilidade de um material de qualidade e padrão profissional no mesmo.

Para tal dividimos nosso percurso de pesquisa em três partes que se revelam em três capítulos na presente dissertação, são eles:

Capítulo 1: Fundamentos Técnicos do Desenvolvimento de Jogos Digitais. Compreensão e Definição de Etapas e Conceitos Básicos;

No presente capítulo forneceremos as conceituações sobre o que é jogo, o que é um jogo digital, o que são jogos bidimensionais e tridimensionais, tal como a conceituação das etapas técnicas básicas da produção de um jogo digital como o *brainstorm*

(tempestade cerebral, etapa essa da qual se decide sobre as etapas listadas a seguir), arte de conceito, modelagem tridimensional, textura e design de nível.

Assim, pesquisaremos e conceituaremos as etapas técnicas básicas de produção, afim de um maior esclarecimento e entendimento do pesquisador-leitor da produção de jogo como um todo, sendo obviamente inviável uma discussão aprofundada de *design* de nível em UDK sem antes ao menos mencionar e conceituar as etapas básicas anteriores de produção que são indispensáveis para se chegar ao design de nível, aqui encontra -se nossa justificativa dos conteúdos discutidos no primeiro capítulo.

Capítulo 2: UDK e sua História;

Nessa segunda etapa da pesquisa discutiremos o que é o motor de jogo UDK, como se deu seu surgimento, suas diferenciações e semelhanças para com a *Unreal 3* (que se dão apenas em nível de políticas de pagamento e lançamento de produto), tal como sua história, sua proposta, quais foram os jogos digitais de padrão *Triple A* lançados feitos em UDK e *Unreal 3* (tendo em vista que o potencial de ambas e o poder de processamento é o mesmo e que a *Unreal 3* deu origem a UDK).

Capítulo 3: Tutorial sobre a Produção em UDK;

Na terceira etapa será feito um relato/tutorial de etapas chave principais design de nível de um ambiente navegável em UDK, mostrando etapas como a importação dos objetos tridimensionais para a UDK, montagem do cenário entre outras etapas a serem cumpridas. O conteúdo será demonstrado e explicado através de *prints* e explicações escritas.

Utilizaremos como normatização Apa modificada, a pedido do nosso orientador.

Capítulo 1: Fundamentos Técnicos do Desenvolvimento de Jogos Digitais. Compreensão e Definição de Etapas e Conceitos Básicos

Figura 1: Imagem de Abertura do Capítulo 1

Enquanto objeto da cultura contemporânea (Manovich, 2001) cada vez mais o jogo digital deixa de ser entendido como um brinquedo ou entretenimento, para ser considerado como uma obra de arte[1]. Pesquisadores como Laurel (1993), entendendo o computador como um espaço de representação teatral, Heim (1994), mostrando as propriedades metafísicas do ciberespaço, Murray (2003), inaugurando uma discussão sobre a transposição da cultura ocidental para o computador, na qual jogos digitais

[1] Em 2011, a partir da *National Endowment for the Arts* (EUA), os *games* (jogos digitais) foram declarados oficialmente como obra de arte nos EUA. Mais informações em Rabin (2011A), página 4, N.R.T. 1;

como *Myst* e *Zork* ocupavam o plano central, (Manovich, 2001), Petry (2003) situando os jogos digitais como óperas digitais e como objetos do fazer artístico, situados na perspectiva de uma escultura digital plástica, Schuytema (2008) organizando a metodologia do *design de games* como uma prática, para citar alguns[2], além de pesquisarem o progressivo alcance e expansão do novo objeto da cultura ocidental, o *game* (jogo digital ou videojogo)[3], dão o pontapé inicial em um processo de reflexão que perguntava acerca do estatuto do objeto digital dentro de uma cultura que tende a classificar e ordenar seus objetos dentro de critérios funcionais, finalidades e perspectivas econômicas.

Hoje em dia, o espaço acadêmico reservado aos jogos se expandiu, saindo da qualidade de um produto comercial destinado ao entretenimento para se tornar um objeto de pesquisa e ensino dentro das Universidades. Sendo o jogo entendido como um objeto cultura, um co - fundador da própria cultura (Huizinga, 2001). Cada vez mais encontramos cursos superiores de jogos digitais[4], os quais atendem a uma crescente demanda, tanto no que diz respeito a importância de se analisar as diferentes metodologias de análise de produção, quanto de *desenvolvimento e didáticas de jogos.*

Diariamente, como podemos ver pelo fórum, uma série de estudantes, artistas digitais e pesquisadores de jogos digitais, passam por problemas técnicos na produção de *assets* de qualidade no motor de jogos UDK como já referido em nosso artigo *Organizando os mapas de iluminação dos assets de arte para os motores de jogos: considerações metodológicas para o caso da produção voltada ao motor de jogos UDK*

> "Todos os dias os artistas tridimensionais enfrentam problemas técnicos e conceituais na dura tarefa de produzir recursos de qualidade para os motores de jogos. Mas, quando uma dificuldade combina ambos, tanto os requisitos técnicos como os conceituais é que eles são alertados para a importância de uma atitude e uma disciplina de trabalho organizada metodologicamente. No presente artigo enfocamos um destes momentos comuns que tem exasperado inúmeros usuários do motor de jogos UDK no mundo inteiro e tem sido fonte

[2] Os exemplos poderiam ser enumerados em uma generosa lista, entretanto nos atemos ao mais frequentemente referidos nas discussões dentro do grupo de pesquisas sobre jogos do qual participamos (NuPHG – Núcleo de Pesquisas em Hipermídia e *Games*) com nosso orientador e colegas;

[3] O termo inglês *game* é muito utilizado pelos usuários. Enquanto isso, o MEC sugere o termo jogo digital para caracterizar um jogo de computador e/ou console. Já o termo videojogo é o modo como o termo game tem sido frequentemente traduzido pelos autores portugueses na literatura especializada. Ainda que o termo videogame seja também utilizado por muitos, usuários e pesquisadores, nos últimos tempos ele se encontra em franco desuso;

[4] De acordo com levantamento no emec.gov.br, o número de cursos superiores de Jogos Digitais é de 50, sendo 16 cursos, oferecidos no Estado de São Paulo (informação de fevereiro de 2014);

> para debates em fóruns, artigos e extensa documentação: a importação de recursos de arte que tenham uma apresentação e performance de qualidade profissional" (Petry; Lopes Filho; Pontuschka; Fragoso; Marques; Zansavio, 2012, p. 37 *in* Branco; Silvano e Malfati, 2012).

> "A produção de recursos tridimensionais para motores de jogos envolve um conjunto de conhecimentos de amplo escopo e o seu domínio técnico e conceitual demanda uma curva de aprendizagem considerável. Ele está inserido em um campo que mescla dinamicamente habilidades e competências artísticas e requisitos técnicos. Como muito bem diz Rabin [2012], o modelador 3D profissional é um escultor e um técnico. Ele é um artista e um engenheiro. Uma análise da situação mostra que o domínio dos processos de produção de recursos de arte tridimensional para o motor de jogos UDK exige do designer um grande esforço de formação e a constante busca de atualização dos conhecimentos técnicos relacionados, tanto no que diz respeito à modelagem 3D, bem como aos requisitos estabelecidos pelo motor de jogo. Inúmeros são os exemplos de solicitações de ajuda por parte dos iniciantes nos usos do motor UDK em relação aos problemas de sangramento (bleeding) na produção dos mapas de luz para o Lightmass" (Petry; Lopes Filho; Pontuschka; Fragoso; Marques; Zansavio, 2012, p. 38 *in* Branco; Silvano e Malfati, 2012)

Como já demonstrado em nosso artigo acima referido, as problemáticas de pesquisa referentes à produção de jogos digitais no motor de jogos UDK, são um terreno bastante fértil, o artigo referido, nos fala sobre o problema de sangramento (*bleeding*), na produção de mapas de iluminação, que são problemas de iluminação que produzem uma sombra escura, onde não deveria haver. Esse é um dos problemas que ocorre com frequência, existindo tantos outros. Muitos profissionais da área de jogos digitais, professores e pesquisadores do motor de jogos UDK, disponibilizam material bibliográfico destinado à didática do mesmo. Podemos citar Finch (2013) autor de The Unreal Game Engine: *A Comprehensive Guide to Creating Playable Levels*, Thorn (2011), com sua obra *UDK Game Development*, Moore (2011) autor de *Unreal Development Kit: Beginner´s Guide*, Mooney (2012) *Unreal Development Kit Game Design Cookbook*, Busby; Parrish; Wilson (2009) *Mastering Unreal Technology I e II* todos referidos livros de ensino do motor de jogos UDK.

No presente capítulo será realizado um levantamento de conceitos técnicos das etapas básicas de produção de um jogo, ou seja, por mais que a presente pesquisa se foque na produção de jogos em UDK, ou seja, especificamente na etapa de *level design*, por mais que as outras etapas não sejam aqui detalhadas serão explicadas conceitualmente. Assim pesquisaremos e explicaremos os conceitos das etapas técnicas de produção, afim de um maior esclarecimento e entendimento da produção de jogo como um todo, sendo inviável uma discussão aprofundada de *level design* em UDK sem antes ao menos mencionar e conceituar as etapas anteriores de produção que são indispensáveis para se chegar ao *level design*. Para tanto conceituaremos etapas como *brainstorm*, *concept art*, modelagem 3D, texturas e o próprio *level design*. Tal como

também torna-se impossível falar de jogo sem que antes se conceitue o que é jogo, jogo como elemento da cultura (Huizinga, 2001), jogos bidimensionais e tridimensionais entre outros conceitos básicos ontológicos e técnicos de produção do qual o pesquisador/leitor deve passar antes de chegar ao *level design* em UDK propriamente dito, para tanto explicitaremos essas questões no Capítulo 1 da presente dissertação.

1.1 - Definição de Jogo

O jogo está no mundo desde as fundações da cultura, podendo ser considerado como um co-fundador da própria cultura (Huizinga, 2001), Johan Huizinga [5](1872-1945) em *Homo Ludens – O Jogo como Elemento da Cultura*, define o jogo como um elemento cultural, e nos define jogo como:

> "O jogo é uma atividade livre, ficando conscientemente tomada como "não séria" e exterior à vida habitual, mas ao mesmo tempo capaz de absorver o jogador de maneira intensa e total. É uma atividade desligada de todo e qualquer interesse material, com a qual não se pode obter nenhum lucro. Ela é praticada dentro dos seus próprios limites de tempo e de espaço e de acordo com regras fixas e de uma maneira ordenada. Promove a formação de agrupamentos sociais, que tendem a se cercar de sigilo e sublinhar a sua diferença em relação ao mundo comum, por disfarce ou outros meios" (Huizinga, 2001, p. 13)

Huizinga nos coloca então, que o jogo:

- É uma atividade livre;

- Exterior a vida habitual e ainda sim capaz de proporcionar imersão;

- Não deve ser relacionada a lucro;

- Acontece dentro de um determinado espaço;

- O jogo possui regras;

- O jogo pode criar grupos;

Em *Design de Games: Uma Abordagem Prática*, Paul Schuytema (2008, p. 7), define jogo digital como:

[5] Johan Huizinga foi um filósofo, historiador e antropólogo conhecido por seus trabalho sobre a Idade Média e tem como uma de suas obras mais significativas *Homo Ludens – O Jogo como Elemento da Cultura*, onde o autor discute a questão do jogo como um elemento co-fundador da própria cultura;

> "Uma atividade lúdica composta por uma série de decisões, limitada por regras, e pelo universo do próprio game, que resultam em uma condição final. As regras e o universo do game são apresentados por meios eletrônicos e controlados por um programa digital. As regras e o universo do game existem para proporcionar uma estrutura e um contexto para as ações de um jogador. As regras também existem para criar situações interessantes com o objetivo de desafiar e se contrapor ao jogador. As ações do jogador, suas decisões, escolhas e oportunidades, na verdade, sua jornada, tudo isso compõem a "alma do game". A riqueza do contexto, o desafio, a emoção e a diversão da jornada de um jogador, e não simplesmente a obtenção da condição final, é que determinam o sucesso do game."

O autor de *Design de Games – Uma Abordagem Prática*, também nos coloca que o jogo é uma atividade lúdica, que ele é limitado por regras, tal como Huizinga nos coloca, porém como a definição de Schuytema (2008, p. 7), é especificamente a de um jogo digital, possui outros elementos específicos do mesmo, como quando nos diz que *"As regras e o universo do game são apresentados por meios eletrônicos e controlados por um programa digital"*, característica inerente à de um jogo digital, ambas as definições, concordam que o jogo é uma atividade voluntária, quando Schuytema nos coloca que é uma atividade lúdica, e ambos concordam que o jogo deve possuir regras.

Novak (2010, p. 186) nos coloca que todos os jogos digitais possuem regras, que mesmo os primeiros jogos de fliperamas possuíam regras mesmo que simples e instruções prévias.

Rabin (2012, p. 58 e 59) nos coloca que jogo é diversão, que tem o propósito de entreter e que precisa ser sem propósito aparente, "*jogar é apenas jogar*", voluntário, diferenciado de comportamentos ou intenções sérias, divertido, seguro e improvisado.

"Os jogos facilitam o ato de jogar, no qual regras ou objetivos ou ambos impõem algum nível de estrutura para as interações" (Salen, 2004, *apud* Rabin, 2012, p. 58)

As definições de jogo dos autores citados, por mais que apontem elementos diferentes por vezes, todos concordam que o jogo é uma atividade lúdica, livre, espontânea, com finalidade de divertimento, que possui regras próprias e um objetivo.

1.2 – Jogos Bidimensionais (2D) e Tridimensionais (3D)

Os jogos bidimensionais são aqueles dos quais não podemos rotacionar a câmera, já os tridimensionais, são aqueles dos quais podemos rotacionar a câmera e assim observamos todas as dimensões de um objeto tridimensional.

Rabin (2012, p. 34), nos define que grande parte dos jogos do estilo Plataforma[6], são em 2D (citando exemplos como as franquias *Sonic* e *Mario*), embora o gênero plataforma também, tenha se expandido para o 3D, o autor também nos coloca que é bastante comum jogos de Luta[7] em 2D e cita exemplos como *Mortal Kombat*.

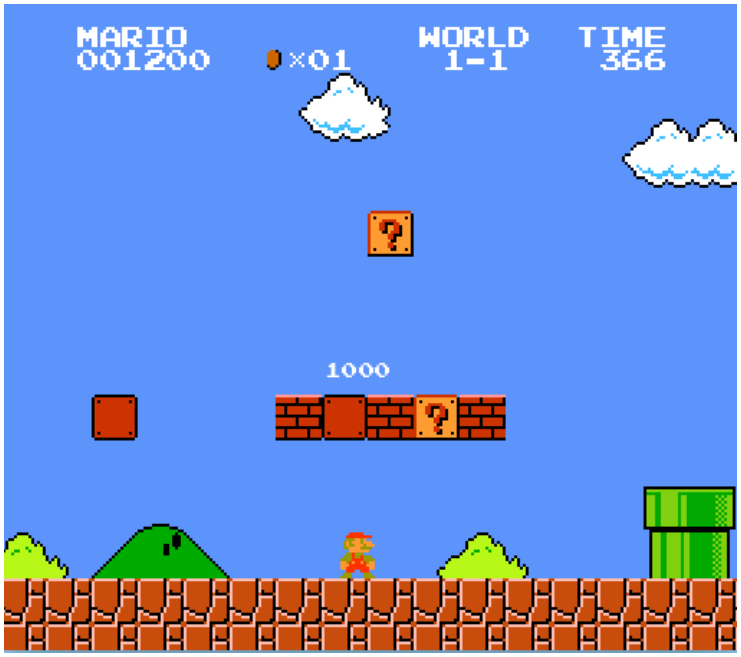

Figura 2: Screnshot de Mario World (1990), exemplo de jogo 2D Plataforma

[6] Rabin (2012, p. 34) define o Estilo Plataforma como: "Os jogos originais de plataforma envolviam a personagem correndo e pulando em um campo de jogo com visão lateral";
[7] Rabin (2012, p. 34) define o Estilo Luta como: "Em jogos de luta, o usuário luta contra outros jogadores ou contra o computador com artes marciais ou espadas. Esse gênero se originou nos fliperamas";

Figura 3: Screenshot de Mortal Kombat 3 (1995), exemplo de jogo bidimensional de luta

São jogos dos quais a personagem pode se movimentar apenas em dois sentidos, para frente e para trás e possui apenas visão lateral.

Já os jogos tridimensionais, são aqueles dos quais podemos nos movimentar no jogo em todas as direções, não só apenas para frente e para trás e podemos ver os objetos nos três eixos X, Y e Z, ou seja, em suas três dimensões por isso objetos tridimensionais. Como podemos ver nas imagens abaixo, os exemplos a seguir de *Bioshock Infinite* (2013) e *Tron* (2010), jogos em 3D feitos no motor de jogo UDK.

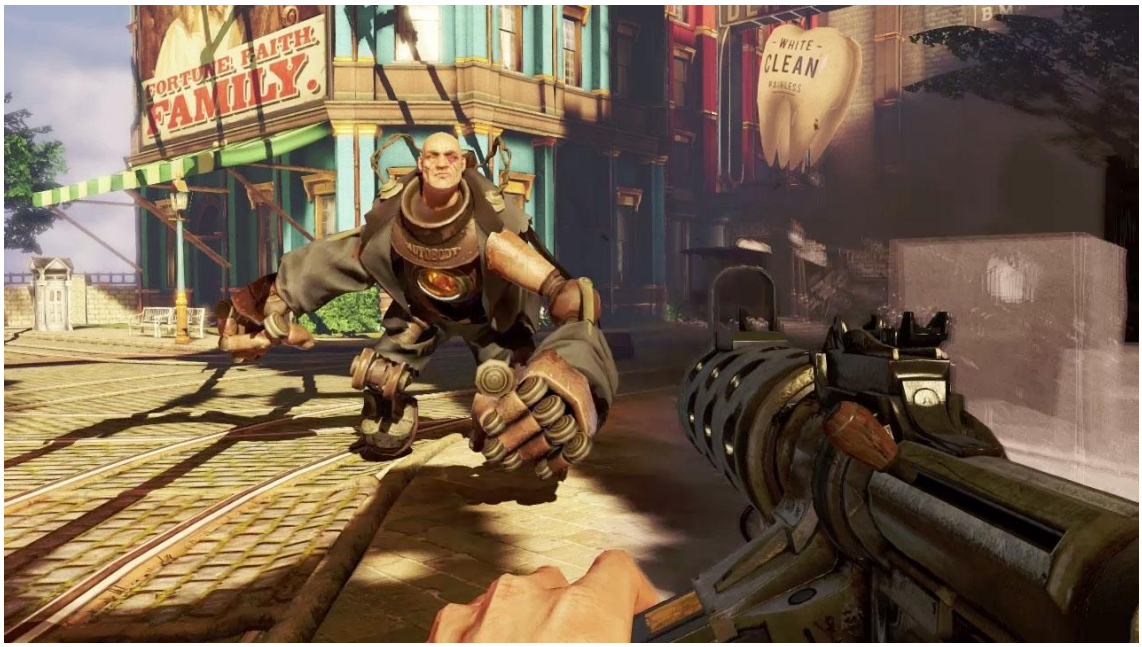

Figura 4: Screenshot de Bioshock Infinite (2013), exemplo de jogo 3D

Figura 5: Screenshot de Tron (2010), exemplo de jogo em 3D

1.3 – As Etapas de Desenvolvimento de um Jogo Digital

A construção de um jogo digital envolve diversas etapas de inúmeras áreas do conhecimento humano, sendo um processo de extrema interdisciplinaridade, abrangendo assim profissionais específicos e especializados para cada etapa do desenvolvimento. As áreas principais são as de roteiro e narrativa, arte de conceito, modelagem tridimensional, texturização, animação, *design* de nível e programação.

Schuytema (2008, p. 12 e 13), discute o ciclo de desenvolvimento de um jogo digital, dividindo primeiramente a produção do mesmo em três principais momentos, são eles: *pré produção*, *produção* e *pós produção*. A *pré produção*, seria onde a equipe de desenvolvedores, criam o conceito do jogo, nessa etapa é feito o *brainstorm*, o roteiro do jogo e narrativa são definidos, é definido o guia de estilo do jogo e então começa a produção da arte de conceito. Em seguida é feito um documento de *design* (conhecido como GDD), contendo as características principais do jogo, esboços, arte de conceito, informações técnicas entre outros, servindo para orientar toda a equipe no processo de produção. Schuytema também colocará que nessa etapa, verifica-se se a equipe possui todos os *softwares* que serão utilizados.

A *produção* é o momento onde o jogo em si é construído, no qual os modeladores/artistas, realizarão a modelagem tridimensional dos cenários e personagens, texturização dos mesmos, em seguida farão a implementação dos mesmos em um motor de jogo, ou seja, construirão o *design* de nível, os *game designers* fazem o que Schuytema chama de roteiro de *gameplay*, ou seja, definem a mecânica do jogo. Em seguida a equipe de programadores define, revisa e implementa o código de programação.

A pós-produção inicia-se quando o jogo digital é lançado, e consiste em produção de conteúdo extra para *download*, possíveis atualizações, se for o caso. Schuytema nos deixa claro (2008, p. 12), que esse esquema é bastante maleável, variando de uma equipe de produção para outra, podendo haver alguma diferença em alguns aspectos, porém em essência, esse é o processo, algumas características não mudam, como por

exemplo, o *concept art* sempre estará presente na pré-produção, tal como a construção do roteiro.

Discutiremos aqui, a definição das etapas de cunho técnico principais contidas em cada um dos três momentos citados (pré-produção e produção, pois os elementos descritos na pós produção são opcionais, ou seja, não é todo jogo que possui conteúdo adicional para download e também muito variável.), sendo estas: *brainstorm*, arte de conceito, modelagem tridimensional, texturização e *design* de nível.

1.3.1 – Brainstorm

O *brainstorm* (termo que traduzido seria tempestade cerebral) funciona como uma tempestade de ideias é o momento do qual a equipe de *game designers* se reúne para uma sessão de ideias, da qual os participantes discutem uma série de aspectos como conceito do jogo, roteiro, guia de estilo, mecânicas, *softwares* a serem utilizados, questões técnicas, entre outros, todos os participantes são convidados a expor suas ideias, cada ideia é anotada e discutida entre todos, sendo validada ou não.

É um momento de intensa criatividade, muito útil na produção do conceito de um jogo ou para uma solução de problemas. Tendo como objetivo que a equipe libere sem restrições toda a sua criatividade e ideias.

> "As sessões de brainstorm são uma oportunidade de envolver a equipe na discussão de várias ideias sobre o jogo. Você pode fazer um brainstorm sobre o conceito inicial do jogo, a jogabilidade básica, o ambiente do jogo ou a aparência dos personagens. Sessões de brainstorm bem gerenciadas também são um ótimo exercício de construção de equipes porque permitem que todos deem suas opiniões sobre o que torna um jogo divertido. (...) Após as ideias serem geradas, passe algum tempo com a equipe agrupando-as em ideias afim e então priorize-as. Partindo disso, gere um relatório dos resultados das sessões de brainstorm e adicione-o às notas da reunião. As ideias de prioridade mais altas serão atribuídas a pessoas específicas para que deem prosseguimento e pesquisem. (...) Agende a sessão de brainstorm como uma das principais tarefas da pré produção para poder ouvir as ideias de todos abertamente. Tente ter o máximo possível de sessões de brainstorm feitas e documentadas na primeira semana do projeto. Quanto mais você adiar a sessão de brainstorm, mais demorará para determinar o conceito inicial do jogo" (Chandler, 2012, p. 217 e 218).

Como vemos pela definição de Heather Maxwell Chandler em *Manual de Produção de Jogos Digitais*, o *brainstorm* é um momento de intensa criatividade e ideias, indispensável na pré-produção de um jogo, para que se definam os conceitos do mesmo. Além disso, o *brainstorm* extrapola para uma maior interação entre os *game designers*, para que todos tenham a oportunidade de expor suas ideias. A autora também nos fala da importância de que tudo seja documentado, para não se perder nada, e que para que a *posteriori*, integrantes da equipe façam pesquisas específicas dos tópicos abordados do conceito do jogo.

1.3.2 – Arte de Conceito

O termo arte de conceito (*concept art*) *consiste* no projeto esquematizado anterior ao seu fazimento, na forma de uma ilustração, para assim se ter a noção final do produto e roteiro de planejamento (assemelhando-se muito com o que Leonardo da Vinci fazia ao esquematizar suas invenções no papel). Ele culmina na produção de conhecimento para o processo artístico e metodológico na produção dos jogos digitais (*games*), o qual é normalmente visto, como o alicerce e ponto de partida para a construção de qualquer jogo com qualidade.

> "Desenvolver um conceito para um novo jogo é como fazer um esboço. Uma ideia é tirada de algum estado precoce e transformada em algo mais elaborado. Mediante esse processo, os detalhes são trabalhados e o conceito se torna mais "real"." (Rabin, 2012ª, p. 119)

> "Quando os concept art são aprovados, a sua arte será a referência de condução e inspiração para um grupo qualificado de modeladores e animadores. Eles vão usar o software de modelagem 3D para criar ambientes e criaturas para habitá-los, tudo isso baseado nos desenhos de conceito" (Stoneham, 2010, p. 59)

A arte de conceito [8] influencia efetivamente a produção da modelagem tridimensional, sua conceitualização histórica e sua presença sistemática em autores e desenvolvedores de jogos digitais. Considerando a arte de conceito como o conjunto de

[8] Ainda que autores como Cardoso (2008) aparentem relacionar o conceito de arte de conceito com o papel do inventor dentro da indústria em geral, em nosso meio, a fixação do conceito ainda não foi realizada.

atividades e procedimentos metodologicamente orientados[9], os quais tem a missão e finalidade de servirem como orientação para o desenvolvimento da totalidade do aspecto visual e estético do jogo, o que os autores americanos chamam às vezes de "*Look and Feel*" [10].

Buscando uma referência histórica, encontramos já no Renascimento, Leonardo da Vinci (1452-1519) que já utilizava da arte de conceito, ao desenhar suas máquinas voadoras Zollner (2005). Diversos autores, como Les Pardew (2004 e 2005), por exemplo, ressaltam a importância da arte de conceito prévia no desenvolvimento de um jogo digital, bem como de todo o processo de *storyboard*, anterior a produção. Bobany (2007) descreve a pré-produção, como a etapa onde toda a arte conceitual será executada, anterior à produção do jogo, a arte de conceito, os detalhes do *design*, para só posteriormente ser produzido na modelagem tridimensional. Stoneham (2010), ressalta durante todo o seu livro (o mesmo se trata disso) *How to Create Fantasy Art fo Videogames* a importância de se produzir a arte de conceito e Weye Yin (2011), descreve a produção da mesma e de toda a composição artística da cena como um todo, como um importante passo anterior a produção tridimensional, em jogos digitais e também em filmes e animações. Rabin (2012), também descreve a arte de conceito como etapa de produção anterior a modelagem tridimensional. É neste sentido, a partir da detecção da importância dada ao tema pela literatura estrangeira especializada, artística e técnica (Pardew, 2004 e 2005; Bobany, 2007; Stoneham, 2010; Yin, 2011 e Rabin, 2012).

Enquanto a arte de conceito no trabalho histórico do Mestre Leonardo, buscava guardava informações para projetos a serem desenvolvidos pelo artista/cientista criador, atualmente as obras cinematográficas e jogos digitais, oferecem em formato literário, detalhados registros visuais e escritos de sua arte de conceito. Hoje facilmente podemos

[9] São as atividades e procedimentos sequencias considerados aqui os a seguir: *key concepts*, arte conceitual, *storyboards*, esboços para *cutscenes*, arte de conceito de *equipamentos e utensílios do game*, arte de conceito de personagens, *de NPCs*, de acordo com (Stoneham, 2010).

[10] È o caso da recente edição do livro de Rabin, Steve (2013). *Introdução ao desenvolvimento de jogos: criação e produção audiovisual*, no qual dedica um capítulo inteiro ao conceito de *design visual* e uma seção dele ao *look and feel*, traduzido para o português como *a "aparência"*.

encontrar inúmeros livros de arte de conceito a venda, tanto de filmes, quanto de animações e de jogos[11].

1.3.3 - Modelagem Tridimensional

Uma das etapas mais importantes da produção de um jogo digital é a modelagem tridimensional, a partir dela é que teremos os cenários e personagens que compõem todo o universo de um jogo digital.

Para tal, o artista tridimensional necessita de um modelador tridimensional, isto é, um *software* utilizado para a produção de modelos tridimensionais. Há uma grande variedade de modeladores tridimensionais disponíveis, como por exemplo, *Autodesk Maya, Autodesk 3D Studio Max, Cinema 4D, Blender*, entre outros.

> "Um modelador 3D profissional é um escultor e um técnico. Ele é um artista e um engenheiro. Ele deve estar preocupado com a forma, a expressividade e estilo, bem como com a contagem de polígonos, a topologia e a eficiência de seus modelos. Embora existam muitos métodos e tipos de modelagem, a modelagem de polígonos nos jogos hoje é a principal." (Rabin, 2013, p.657).

Como podemos ver o modelador não pode se dar ao luxo de simplesmente modelar de forma intuitiva, é necessário que se pense na quantidade de polígonos dependendo do projeto que se tem em mente e do motor de jogo que se vai usar, se o jogo for feito em um motor de jogo potente como a UDK ou a *Unreal 3*, a preocupação com polígonos de fato não precisa ser alta, pois o motor de jogo do qual utilizamos em nossa presente pesquisa suporta uma grande quantidade de polígonos, tendo sido utilizado em grandes produções como mostraremos a *posteriori*[12].

Neste sentido, podemos pensar nesses passos metodológicos (Petry, 2003), como uma atitude de cunho ontológico onde essa modelagem tridimensional é pensada, construída e organizada metodologicamente de uma forma didático - acadêmica.

---

[11] Portais, Blogs e Organizações novos e interessantes são encontrados na Web a cada dia. Exemplo disso é a Ong *Concept art* (http://www.conceptart.org/forum.php#.UVXM7hyG18E), que reúne artistas e técnicos dedicados ao tema do *concept art*. Uma simples pesquisa no site de livros *amazon.com*, com as chaves de busca *concept art* produz um retorno de mais de mil publicações diretamente relacionadas ao tema.

[12] No capítulo 2 da presente pesquisa: A História do Motor de Jogos UDK;

Quando se trata de modelagem tridimensional para o motor de jogos UDK, a principal preocupação é a realização do *lightmap*. Uma série de *assets* de arte, quando importados para a UDK, apresentam problemas quando submetidos ao sistema de *Lightmass* da UDK, problema de "sangramento", para que esse problema não ocorra, os modeladores tridimensionais têm de acatar em suas produções, uma série de requisitos na confecção do mapa de iluminação, conhecido como *lightmap*, para seus modelos tridimensionais, como já discutimos em nosso artigo *Organizando os Mapas de Iluminação dos Assets de Arte para os Motores de Jogos: Considerações Metodológicas para o Caso da Produção Voltada ao Motor de Jogos UDK*, no qual demonstramos o referido processo e damos a seguinte definição de como ele se dá:

> "A Parametrização do mapa de Iluminação no Maya 2012: O mapa de iluminação utilizado pelo UDK para o *Lightmass* se constitui em uma projeção ortogonal da superfície total do objeto disposta em um plano que segue coordenadas XY [Wright, 2009]. Como mapa de iluminação, o UDK utiliza um canal de UV do objeto importando, por default, no canal de Um de UV do objeto. Ele é armazenado em uma variável no painel de Edição do Objeto, designada por Light Map Coordinate Index. Para o mapa de texturização, o UDK reservará o canal 0 (Zero) das UVS do objeto. (Petry; Lopes Filho; Pontuschka; Fragoso; Marques; Zansavio, *in* Branco; Malfatti; Lamar, 2012, p. 92)

Podemos ver na imagem abaixo (retirada do nosso tutorial citado acima) um objeto tridimensional com o chamado *bleeding*, sangramento, para melhor ilustrarmos:

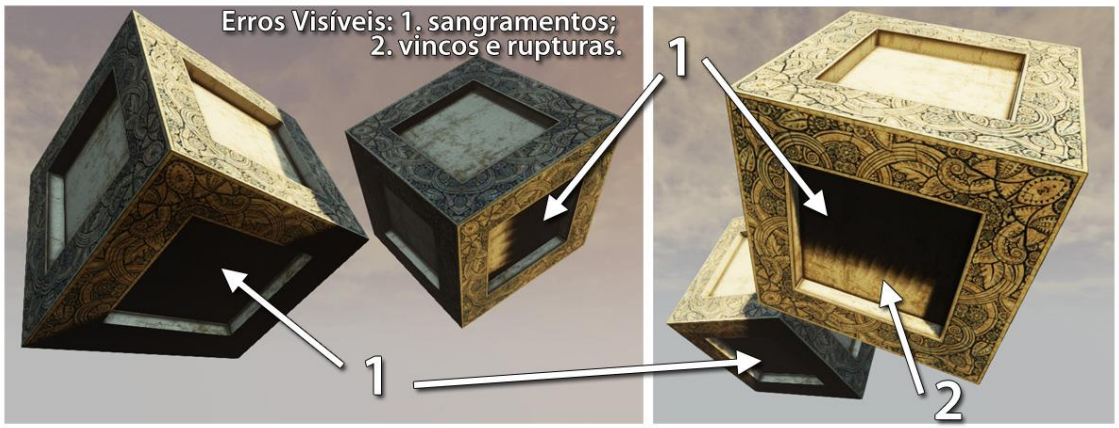

Figura 6: Cubo Metafísico com problemas de sangramento (Petry; Lopes Filho; Pontuschka; Fragoso; Marques; Zansavio, *in* Branco; Malfatti; Lamar, 2012)

Para que o demonstrado na imagem acima não ocorra, deve-se realizar corretamente a modelagem tridimensional seguida do mapeamento UV (sendo esse o manual e não o automático se fizer automático o erro acima pode ocorrer) e do *lightmap*.

Assim, esse canal de UV produzido no *software* de modelagem, será utilizado para a produção do mapa de iluminação do objeto a ser exportado para o UDK, correspondendo ao segundo canal criado no Modelador *Maya*.

Como podemos ver abaixo a correspondência entre os canais:

UDK		**Maya**
Canal 0	texture	Canal 1
Canal 1	mapa de iluminação	Canal 2

Quando a modelagem, o mapeamento UV e *lightmap* do objeto tridimensional são feitos corretamente, apresenta-se um resultado de alta qualidade, como podemos ver na imagem abaixo:

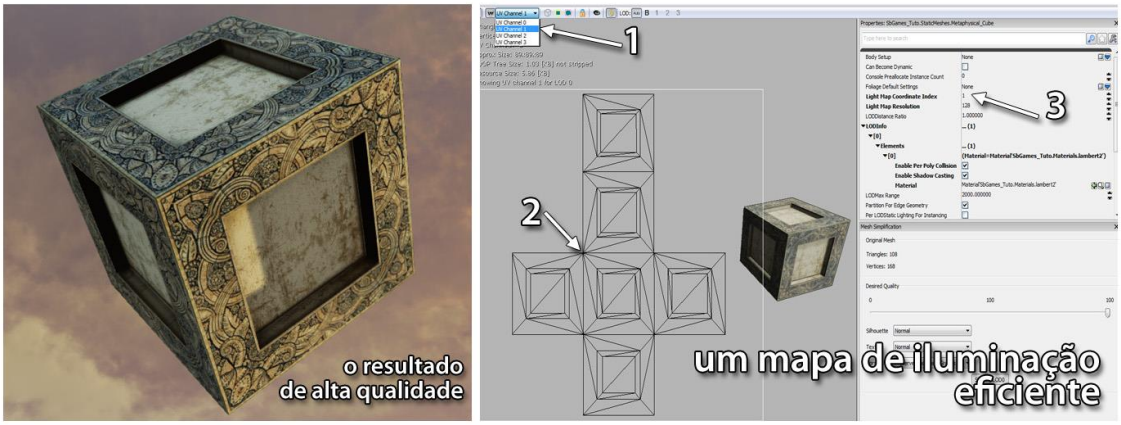

Figura 7: Cubo Metafísico com mapa de iluminação correto (Petry; Lopes Filho; Pontuschka; Fragoso; Marques; Zansavio, *in* Branco; Malfatti; Lamar, 2012).

Rabin (2013, p. 676 e 677) define uma metodologia básica para o desenvolvimento da modelagem tridimensional, explicitamos a seguir:

- ***Uso de Materiais de Referências:*** Quanto maior for a diversidade e a quantidade de materiais de referência para a retratação de um objeto, ambiente etc. melhor e mais fidedigno será o resultado da modelagem tridimensional. Rabin também sugere aqui que se utilize um desenho contendo diversas visões daquilo que será modelado, de frente, de lado, atrás e de cima. Também sugere a criação de texturas a partir do material de referência.

- ***Trabalho Inicial e Refinamento:*** Consiste em iniciar a modelagem por formas e blocos primitivos para as estruturas principais ou formas maiores ou mais detalhadas, pois afirma que é mais fácil a identificação de eventuais problemas do modelo tridimensional no início da modelagem, resolvendo assim os maiores problemas identificados, ainda na fase primária da modelagem do objeto, para em seguida dar um maior detalhamento à peça tridimensional.

Podemos perceber claramente a questão do conceito do *pensar e fazer* já apontada por Richard Sennett em *O Artífice* na modelagem tridimensional para jogos digitais, o conceito remete as oficinas do período artístico renascentista, na qual os artesãos em seus ateliês eram tomados como artistas. Sempre correlacionando os conceitos do fazer e pensar, colocando-nos que fazer é pensar, podemos pensar no conceito do *Artífice* na questão do modelar para jogos digitais, como já colocado e proposto por nós no artigo *Wasteland Beautiful: O Estatuto Ontológico nos Mundos Tridimensionais dos Games,* como podemos verificar abaixo:

> "Sennett (2009) apresenta o conceito de artífice e o aplica em uma discussão sobre as relações cooperativas entre o pensar e o fazer, entre cérebro e mão. A oficina do artífice se constituiria em um espaço de produção ontologicamente privilegiado. Tomando como modelo de sua visada, a oficina renascentista e a transformação do artesão em artista, poderia ser elucidativa para entendermos os atuais ambientes de trabalho em games que funcionam ao modo de oficinas (Sennett, 2009: 67-96). Para pensarmos a imagem digital, o conceito de artífice é fundamental. Ele nasce de um desafio lançado pelo filósofo Richard Foley que, em um momento de impasse reflexivo de Sennet lhe perguntou: Qual intuição que o orienta? E Sennett respondeu: fazer é pensar. Da interrogação do colega nasceu um programa de pesquisa, o qual produziu uma problematização que chegou a nós pelo conceito de artífice. Pensamentos e sentimentos estão inclusos no processo do fazer. No fazer, o homem pensa com os materiais, pensa e sente em meio às coisas, enfrentando o dilema sempre problemático que se abriga no fazer, entre o como, o quê, o porquê e, também, o para quê ou quem. Sennett nos diz que para que possamos aprender com as

> coisas precisamos aprender a apreciar as suas qualidades. Para apreciarmos um desenho ou uma pintura, anteriormente temos de aprender a apreciar o traço, a hachura, a cor e a pincelada. No centro da questão do artífice, Sennett encontra a habilidade artesanal, um conceito que até então parecia estar esquecido e soterrado pela ação da técnica, o qual ele irá resgatar, do renascimento aos estúdios dos modeladores tridimensionais e dos pintores digitais." (Petry; Costa; Silva; Marques; Casarini, 2013, p. 75 e 76 *apud* Sennett, 2009. Disponível em: http://revistesdigitals.uvic.cat/index.php/obradigital/article/view/26/37, Acessado em 04/2014).

1.3.4 – Texturização

A texturização é o processo do qual dará cor e realismo aos objetos modelados, trata-se de um processo muito parecido com a corriqueira tarefa de embrulhar um presente, essa informação pode ser melhor exemplificada a partir da seguinte definição:

> "Muito parecido com cobrir um livro com uma capa de papel ou embrulhar uma bola de futebol em papel ou a pintura de uma escultura de um cavalo em cerâmica com tintas esmaltadas, da mesma forma um modelo 3D em um jogo precisa de algo equivalente a uma cobertura, um embrulho ou um revestimento de cor" (Rabin, 2013, p. 687).

O processo de texturização possui várias etapas, a primeira delas é a fabricação das imagens bidimensionais pelo artista digital, o artista criará ou modificará essas imagens em um *software* de edição de imagem 2D, como o *Photoshop* por exemplo, criando assim o mapa de textura.

O artista digital deve preparar a textura da forma mais adequada e melhor uso em relação à geometria do modelo tridimensional (Rabin, 2013, p. 687). Esse processo recebe o nome de mapeamento UV, tal como a produção do *lightmap* que já foi discutida e demonstrada no item anterior.

De acordo com Rabin (2013, p. 689 e 690) existem três tipos de mapas de textura baseados em arquivos de imagens bidimensionais, são eles:

Mapa de cores: Trata-se da textura que fornece as cores para um modelo tridimensional, podendo essas serem compostas de cores básicas ou difusas, serem

hiper-realistas ou produzidas através de pintura digital, tudo dependendo do guia de estilo definido para o jogo;

Mapa de transparência: São mapas compostos por uma imagem a cores ou em escala de cinza, usando-se combinadamente um mapa de cores para o controle da transparência. Também conhecidos como mapas de opacidade ou canal *alpha*, é largamente utilizado para fazer folhagens de diversos tipos, grades, entre outros;

Mapas de Bump: O *bump map*, é o responsável pela aparente profundidade e relevo que vemos nas texturas dos objetos nos jogos digitais, responsável pela rugosidade, relevo e sombreamento na aparência dos objetos tridimensionais. Sendo de grande importância para uma aparência mais realista e convincente nos jogos digitais; Como podemos ver na imagem abaixo um muro sem o *bump map* e outro com o *bump map*;

Figura 8: O Cubo Metafísico com Bump e sem Bump. Produzido por Gabriel C. Marques

Podemos concluir aqui que a textura é que da vida aos modelos tridimensionais e ambientes de um jogo, a textura da a cor, o acabamento, a noção de profundidade e de vincos com o *bump*. Sem a textura, os modelos teriam a cor original cinza dos modelos tridimensionais no modelador 3D (antes de receberem a textura), pode-se dizer que sem textura não há jogo, trata-se de uma etapa de extrema importância da qual define

seriamente a qualidade da experiência estética (Bateman, 2012), tal como na qualidade gráfica e atmosfera crível do jogo.

1.3.5 – Design de Nível

A etapa do *design* de nível (*level design*) consiste na montagem das várias etapas do jogo em si, ou seja, na implementação dos objetos tridimensionais (*static mashes*) que compõem os cenários, tal como na implementação da programação (seja ela em códigos, visual ou ambas, dependendo do motor de jogo) que torna possível a jogabilidade e a interação. Novak (2011) coloca que normalmente a equipe de *design* de nível trabalha dividindo a produção de acordo com os "níveis" ou "fases" do jogo, trabalhando-as uma por vez, sendo muito importante questões como quantos níveis terá o jogo, a curva de dificuldade de cada um (chamado de *flow*), o objetivo de cada nível do jogo, entre outros aspectos. Novak (2011) define *design* de nível como:

> "Design de Nível é definido como a criação de ambientes, cenários e missões em jogos eletrônicos. O level designer usualmente utiliza de ferramentas de design de nível (ou level editors), tal como Valve Hammer Editor (Valve Software), Unreal (Epic Games), World Builder (Electronic Arts), Aurora Toolset (Bioware) (...) Level designers podem usar motores de jogo ou ferramentas autorais como Unity 3D, Torque 3D e GameSalad" (Novak, 2011, p. 214).

A partir do *design* de nível que se criam os ambientes do jogo, os cenários, é a partir dele que os objetivos do jogo são implementados, é basicamente a montagem do jogo, em cada uma das fases ou níveis do mesmo. Claro que para isso possa se dar, necessita-se de que as outras etapas de produção que conceituamos anteriormente estejam prontas, caso contrário não há o que "montar", o que colocar na montagem dos cenários, ou seja, os modelos tridimensionais têm de estar anteriormente prontos e texturizados, e para que os mesmos estejam previamente prontos e texturizados, anteriormente ainda a arte de conceito, os desenhos conceituais, tem de também estarem prontos e para tal anteriormente tem de ter acontecido o *brainstorm*, para que as definições do que será feito, do conceito do jogo estejam definidas, para tanto um etapa precisa da outra. Há uma conversa entre as etapas que se interlaçam uma com a outra, nos remetendo a teoria da complexidade de Edgard Morin, (1990) que nos propõe uma visão de mundo da qual

as coisas não são vistas em partes, individualmente, mas sim como um todo do qual é indissociável, o autor no propõem uma abordagem interdisciplinar e transdisciplinar que é o que vemos aqui nas etapas técnicas básicas da produção de um jogo digital.

Para tanto, tornar-se-ia impossível ou no mínimo desconexo e sem sentido se partíssemos direto para a história da UDK e para o tutorial ou estado da arte da UDK, sem antes conceituar as partes desse todo técnico que é indissociável.

Segue abaixo uma imagem do *level editor* da UDK em branco, é nele que se constrói o *design* de nível de um jogo ou ambiente navegável feito em UDK.

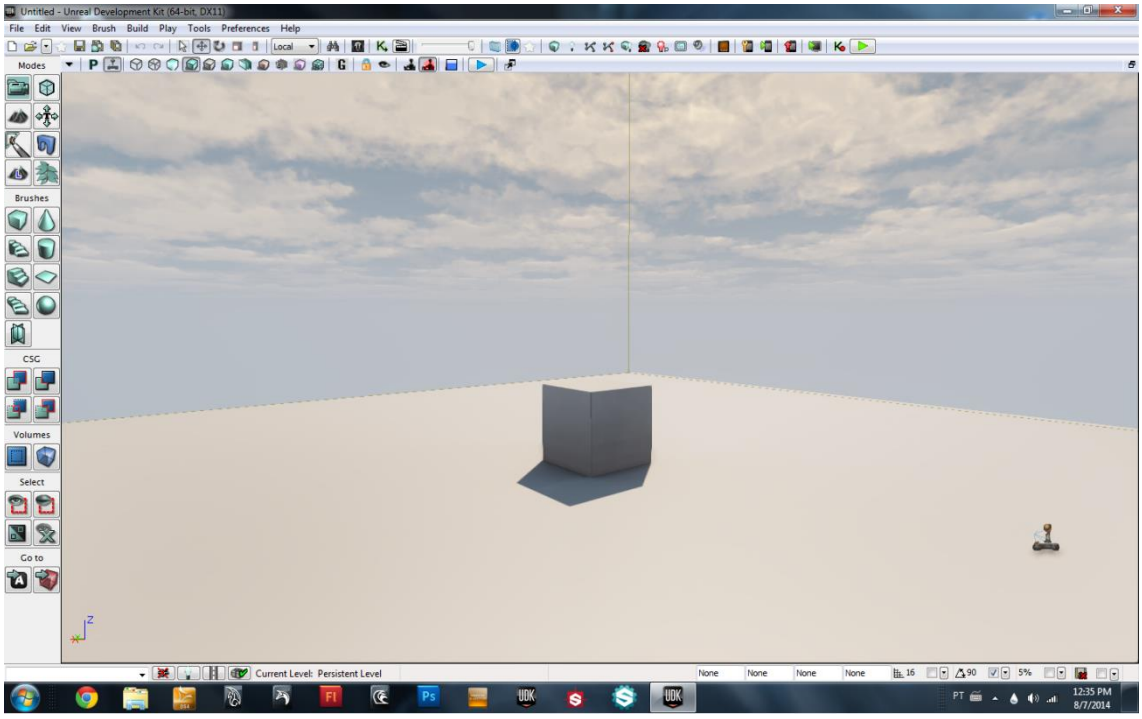

Figura 9: Level Editor da UDK

Também podemos ver na imagem abaixo um *print* do *Content Browser*, trata-se de um recurso da UDK do qual armazena todos os objetos tridimensionais importados para o motor de jogo, a partir do momento que se importa um objeto 3D, ele estará sempre disponível no *Content Browser* para futura utilização, do *Content Browser* passa-se o objeto para o *level editor*.

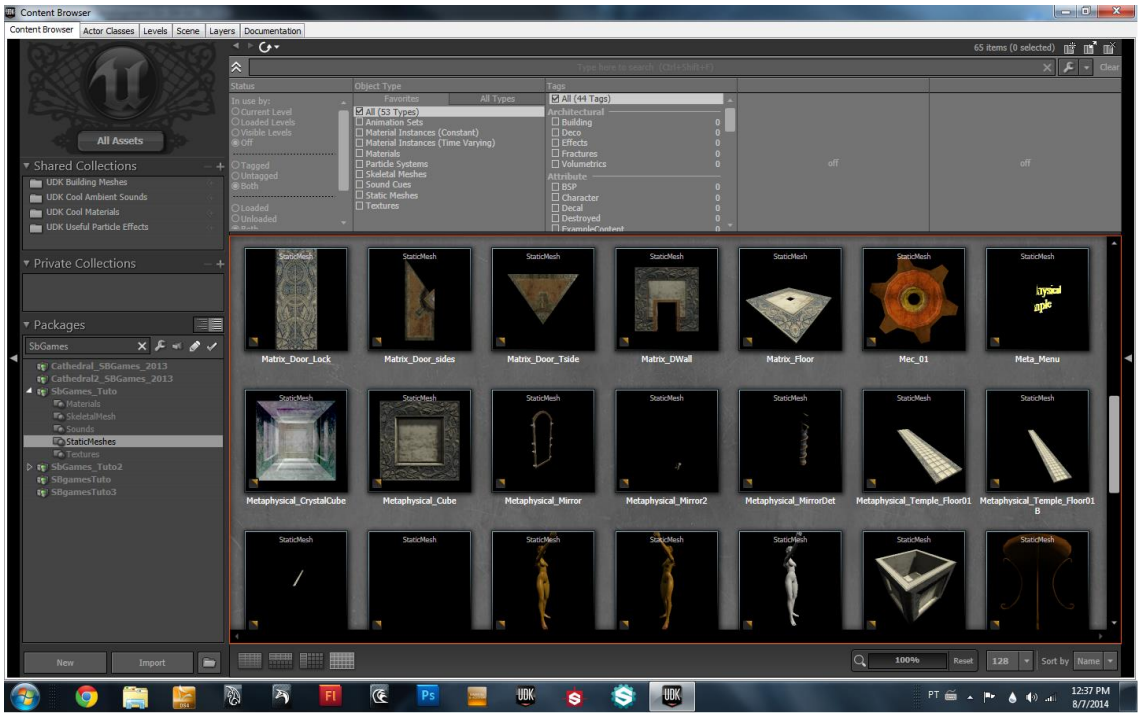

Figura 10: Content Browser

O trabalho com a UDK e assim como em outros motores de jogo, requer um conhecimento introdutório das muitas funções do Editor, janela de efetivo desenvolvimento onde ocorre o *design* de nível. O Editor também é a principal ponte entre as outras funções sumariamente importantes para o desenvolvimento como o *Content Browser* (mostrado na figura 9 acima), local onde fica disponível visivelmente os *assets* e as edições de configuração dos mesmos. O Editor também da vasão para a janela do *Kismet*, onde a partir de uma programação lógica visual você pode desencadear eventos e condições para a evolução do jogo. A janela do *Kismet* é demonstrada na Figura 10, logo a seguir.

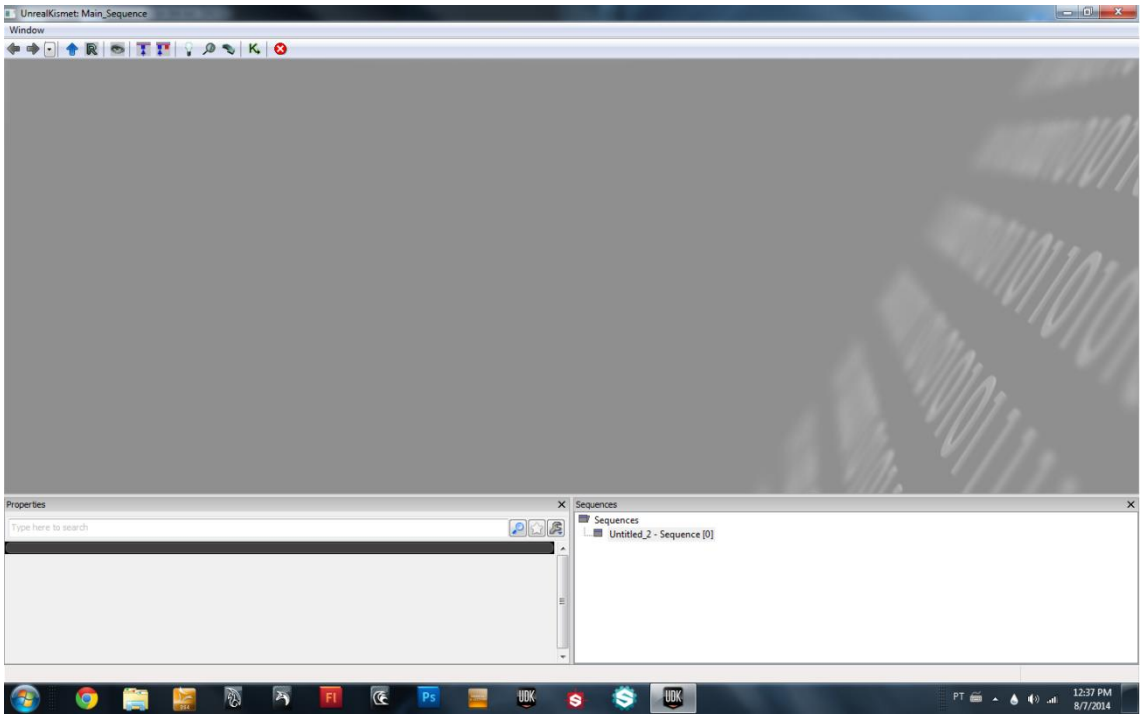

Figura 11: Janela do Kismet

No presente capítulo explicamos as principais etapas de desenvolvimento de um jogo digita para uma maior compreensão do todo. No capítulo seguinte explicitaremos a história do motor de jogo UDK.

Capítulo 2: UDK (Unreal Development Kit) e sua História

Figura 12: Imagem de Abertura do Capítulo 2

Nesse capítulo iremos explicitar o que é a UDK, falar da sua história e surgimento, vantagens e indicações de utilização da mesma e alguns recursos, em seguida explicaremos o que é *Unreal Engine* (notem que é da *Unreal Engine 3* que deriva a UDK), um pouco da sua história e evolução até a derivação da *Unreal 3* para UDK. Em seguida realizamos uma levantamento de jogos digitais de padrão *Triple A* feitos em *Unreal 3* ou UDK, afim de melhor relatar a história do presente motor de jogo e de melhor explicitar suas aplicações, potencialidades e sua posição no mundo dos produtores de jogos digitais.

É de extrema importância que se estamos pesquisando uma ferramenta, não só demonstremos material sobre o desenvolvimento na mesma como será realizado no capítulo posterior, mas que também expliquemos o que é essa ferramenta, seu surgimento, sua história, seus recursos, aplicações, vantagens, política de *royalties*, alguns recursos, potencialidades, tal como um levantamento do que foi produzido (de qualidade) com a mesma. Visando integrar à pesquisa um maior corpo de conhecimento

sobre o material e as questões pesquisadas. Tal como possibilitando ao nosso interlocutor uma melhor visão das potencialidades da ferramenta em si.

2.1 – O que é a UDK

UDK (*Unreal Development Kit*) é um motor de jogo derivado da *Unreal 3*, lançada em 2010, com uma política *free* para estudantes da área de desenvolvimento de jogos digitais. Qualquer indivíduo interessado a partir do ano de 2009 até hoje, pode baixar o *software* UDK e utilizá-lo sem ter de pagar nada a não ser que se lance um produto comercial. Todavia se esse mesmo indivíduo lançar comercialmente um jogo produzido em UDK, ele deverá pagar 25% dos lucros de *royalties*, essa informação consta na EULA (*Engine and User License Agreement*), que aparece disponível para todo indivíduo que baixar a UDK, no momento da instalação do *software*. Tal como também não devem retirar a tela da qual aparece o logo da UDK, ao iniciar o jogo pronto. Uma alternativa muito amigável financeiramente para estudantes, já que os mesmos passavam a ter acesso a uma ferramenta de desenvolvimento atual e largamente utilizada pela indústria de jogos digitais, sem para isso ter de dispor de pagamento.

Figura 13: Logo da UDK

Alan Thorn (2011), em sua obra *UDK Game Development*, nos dá uma definição do que é UDK, como explicitamos abaixo:

> "UDK é a sigla para três palavras: Unreal Development Kit. Em suma, a UDK é um software de criação de jogos. Ou seja, é um conjunto de ferramentas de desenvolvimento, tais como editores, sistemas de script e compiladores, para fazer em tempo real jogos de vídeogame em 3D. A primeira palavra no título, Unreal, não é uma declaração filosófica de que o kit de desenvolvimento não existe. Isto é, isso não significa que o kit não é real. Em vez disso, é o nome do motor de jogo usado para criar jogos com as ferramentas de desenvolvimento do kit. Assim como um motor de carro é usado para alimentar um automóvel, um motor de jogo é usado para alimentar um jogo. Assim, ao mais alto nível, a UDK é composta por duas peças fundamentais e complementares: o Unreal Engine e as ferramentas de desenvolvimento" (Thorn, 2011, p. 4).

A UDK é largamente reconhecida pela qualidade dos jogos produzidos com a mesma, tendo grandes franquias de jogos de padrão de qualidade *Triple A* produzidos na mesma. Discutiremos mais aprofundadamente essa parte logo mais adiante no item 2.3 do presente capítulo.

2.1.1 – As Vantagens da UDK

Thorn (2011) nos enumera as principais vantagens do motor de jogo UDK e sua utilização, são elas:

- ***A UDK é incrivelmente acessível, é rápida e poderosa:*** Quando o autor nos coloca que ela é acessível, nessa parte especificamente ele não se refere à acessibilidade econômica, mas sim acessibilidade de que qualquer pessoa que comece estudar o motor de jogo pode aprender a usar efetivamente suas ferramentas sem conhecimento prévio específicos de programação, também coloca que as ferramentas da mesma, tratam-se de ferramentas mais intuitivas do que de outros motores de jogo.

Aqui o autor também coloca que se trata de um motor de jogo rápido, ou seja, com um bom poder de processamento e da qual podemos ver imediatamente como está o resultado do trabalho, isto é, a UDK possui um comando chamado *Play From Here*, clicando nesse comando, o usuário pode jogar em tela cheia no ambiente que ele está

criando na UDK antes mesmo que ele esteja pronto, muitos motores de jogo como a Unity, por exemplo, não possuem esse recurso. O autor ressalta também o poder desse motor de jogo colocando que ele possui sistema de iluminação indireta incluído, sistema de editor de partículas (as partículas servem para, por exemplo, fazer névoa, chuva dentre outros do tipo), *real time preview*, no caso esse seria o *Play From Here* que falamos logo acima, editores de material (nos editores de material faz-se o material de água dentre outros do tipo), dando ao usuário uma ferramenta completa que possibilita bastante independência.

- ***A UDK suporta vários formatos utilizados pela indústria:*** Isto é, ela suporta PNG, TGA, DAE, FBX, WAV dentre outros. Esse tipo de fator facilita muito o trabalho de um desenvolvedor, fazendo com que ele perca menos tempo convertendo formatos de materiais para importação no motor de jogo.

- ***Programadores podem se focar mais na criação do game:*** A UDK contém tudo o que a *Unreal Engine* (3) contém (a diferença das duas está no sistema de *royalties e que na UDK* o usuário não tem acesso ao *sourcecode* da mesma), isto é, o autor nos coloca que é um motor de jogo completo que poupa e muito o trabalho de um programador, não só porque ela possui um sistema de programação visual chamado *Kismet*, que da conta de uma boa parte do processo de programação, mas também por outros motivos, como por exemplo, ela já ter toda uma estrutura pronta para se construir o jogo, com tantos editores como os de material, partículas e outros, dando assim a oportunidade do programador se focar melhor nos *scripts* do jogo.

2.2 - Unreal Engine, História e Evolução

Unreal Engine é um motor de jogo desenvolvido pela *Epic Games*, tendo tido sua primeira versão em 1998, tendo sido o primeiro jogo digital feito em Unreal desenvolvido pela *Epic Games* e o jogo chamava-se *Unreal* e foi lançado em 1998 e teve uma versão estável finalizada em 2000, chamada *Unreal Tournament*, (franquia

esse existente até hoje, sai um *Unreal Tournament* para cada versão de *Unreal Engine*[13]).

Inicialmente a *Unreal* foi pensada para ser um motor de jogo utilizado preferencialmente para jogos de tiro em primeira pessoa, com o tempo isso foi mudando porque produtoras de jogos começaram a utilizá-la para produzir jogos de diversos estilos. Todavia até hoje a câmera em primeira pessoa já vem pronta no motor de jogo para ser utilizada, mantendo de certa forma um pouco da antiga tradição e propósito.

A *Unreal* foi escrita em C++, e tem como linguagens de programação duas opções para serem utilizadas juntas, são elas:

- ***O Kismet*** (programação visual em fluxogramas);

- ***O Unreal Script*** (linguagem de programação própria baseada em C++);

Esse esquema se manteve até a UDK, na *Unreal 4* a linguagem de programação é o *Blueprint*, baseada em C++.

A *Unreal Engine* teve as seguintes versões:

- Unreal 1: Lançada em 1998;

- Unreal 2: Lançada em 2002;

- Unreal 3: Lançada em 2006;

- UDK: Lançada em 2009;

- Unreal 4: Lançada em março de 2014;

As versões de *Unreal* 1, 2 e 3 possuíam licenças única exclusivamente comerciais com preços altos (por serem versões antigas o preço não consta mais no site da *Unreal*), da *Unreal 3* foi derivada a UDK, esta podia ser baixada livremente e utilizada, todavia se um produto comercial é lançado, tem de se pagar primeiramente 99 (noventa e nove)

[13] Informações disponíveis em: http://en.wikipedia.org/wiki/Unreal_Engine. Acessado em 10/2014. Como se pode ver nas referências utilizadas na página, todas elas eram do site oficial da *Unreal*, porém na reformulação essas informações foram tiradas do ar;

dólares de licença mais 25% de *royalties* dos lucros[14] adquiridos (Essa informação aparece disponível na instalação da UDK apenas).

A *Unreal 4*, já possui um sistema diferenciado das outras quatro versões, com ela o usuário paga mensalmente 19 (dezenove) dólares e se lançar algo comercialmente, paga 5% de *royalties* dos lucros[1].

Notem que mesmo com o lançamento da *Unreal 4*, a UDK não foi descontinuada, tendo tido inclusive atualizações, ainda é a única versão *free* para *download*, tendo assim um forte foco acadêmico.

2.3 – Jogos Triple A Feitos em Unreal 3 ou UDK

Aqui demonstraremos um levantamento realizado dos jogos digitais de padrão *Triple A* (Padrão AAA, de alto padrão de qualidade) produzidos em *Unreal 3*/UDK, notem que seria impraticável colocar todos os jogos produzidos no presente motor de jogo, sendo uma lista gigantesca (centenas), o mais importante é que se assim o fizéssemos nos desviaríamos da proposta aqui que é demonstrar a potencialidade do presente motor de jogo, e a mesma é significativamente demonstrada através de jogos de alto padrão de qualidade, dos quais listamos abaixo, para uma maior compreensão da utilização dessa ferramenta, sua posição importante atualmente na área dos jogos digitais, tal como sua potencialidade. Assim listaremos abaixo o nome dos jogos produzidos e o ano, desses escolhemos 3 (três) com maior expressão de qualidade visual (e por outras razões específicas que detalhamos adiante) para falarmos sobre e colocarmos *prints* dos mesmos, são eles: *Alice Madness Returns* (2011), *Batman Arkham Origins* (2013) e *Bioshock Infinite* (2013).

Lista de Jogos *Triple A* feitos em *Unreal 3*/UDK (listados em ordem alfabética):

- Alice Madness Returns (2011);

[14] Conforme está descrito na EULA (ENGINE END USER LICENSE AGREEMENT). Disponível em: https://www.unrealengine.com/eula. Acessado em 10/2014;

- Batman Arkham Asylum (2009);

- Batman Arkham City (2011);

- Batman Arkham Origins (2013);

- Bioshock Infinite (2013);

- DMC: Devil May Cry (2013);

- Gears of War (2006);

- Gears of War 2 (2008);

- Gears of War 3 (2011);

- Gears of War Judgment (2013);

- Injustice: Gods Among Us (2013);

- Mass Efect 1 (2007);

- Mass Efect 2 (2010);

- Mass Efect 3 (2012);

- Mirror's Edge (2008);

- Mortal Kombat (2011);

- Papo y Yo (2012);

- Silent Hill Downpour (2012);

- Tony Hawk's Pro Skater HD (2012);

- Transformers: Revenge of the Fallen (2009);

- Transformers: War of Cybertron (2010);

- Transformers: Fall of Cybertron (2012);

- Tron: Evolution (2010);

- Unreal Tournament 3 (2007);

- X - Men Destiny (2011);

- X - Men Origins (2009);

- Outlast (2013);

2.3.1 – Alice Madness Returns

Figura 14: Alice Madness Returns

Alice Madness Returns é um jogo digital em 3D, em terceira pessoa, classificado pela ESRB com o gênero Ação/Aventura, multiplataforma, disponível para PC, *Playstation 3* e *XBOX 360*, lançado em 2011 pela *Eletronic Arts*, produzido pela *Spicy Horse* e idealizado e dirigido por American Mcgee, o jogo em questão foi feito no motor de jogo UDK. Escolhemos esse jogo em questão pelo fato de que o mesmo foi produzido por

apenas 13 (treze pessoas) mais o *game designer* American Mcgee, totalizando em uma equipe de apenas 14 (catorze) pessoas, mesmo assim ganhou o prêmio de melhor direção de arte em 2012 sendo considerado um jogo *Triple A* de grande sucesso em vendas, lançado e patrocinado por uma das maiores *publishers* de jogos do mundo. Isso mostra que é possível produzir um jogo digital de grande qualidade artística e gráfica em UDK, mesmo sem uma produção de milhares de pessoas.

O jogo é uma leitura de Alice no País das Maravilhas de Lewis Carrol, é a sequência de *American Mcgee's Alice* (lançado em 2000), se passa em Londres no período vitoriano em uma parte e outra parte dentro da mente de Alice em um País das Maravilhas destruído pelo Trem Infernal. Conta a história de Alice Lidell que perdeu os pais e a irmã em um incêndio, logo após a garota perdeu parcialmente a memória e ficou louca, sendo internada em um hospício. Posteriormente recebeu alta e se encontra em um orfanato chefiado pelo psiquiatra Dr. Bumby, no jogo Alice volta a alucinar após sessões de hipnose e volta ao País das Maravilhas a fim de encontrar suas memórias perdidas e descobrir o que realmente acontece com ela e com as crianças do orfanato descobrindo quem construiu o Trem Infernal.

Figura 15: Alice em Queenslan

O Jogo segundo Mcgee (2011) utiliza de diversos movimentos artísticos como inspiração, sendo eles o surrealismo, o impressionismo e o movimento *steam punk*, também usa da arquitetura gótica.

Figura 16: Alice em Hatters Domain

Todas as imagens são *prints* do *gameplay* do jogo, como podemos ver é possível fazer um jogo de qualidade gráfica superior com o motor de jogo UDK, mesmo sem uma grande equipe para polimento.

2.3.2 – Batmam Arkham Origins

Figura 17: Batmam Arkham Origins

Batmam Arkham Origins (2013) é o mais recente jogo lançado da franquia Batman, é um jogo em 3D, em terceira pessoa, classificado pela ESRB como Ação/Aventura, multiplataforma, disponível para PC, *Playstation 3*, *XBOX 360* e *Wii U*, lançado em 2013 pela *Warner Bross* e produzido pela *Games Montreal*, feito com o motor de jogo *Unreal 3*.

Escolhemos esse exemplo por ser um jogo digital que compõem uma franquia de grande sucesso, por ser um jogo bastante recente (2013) feito em *Unreal 3* (a matriz da UDK), mostrando que trata-se de um motor de jogo extremamente atual e em alta para a produção de jogos digitais. Também escolhemos pelo presente jogo ter sido muito bem recebido pela crítica, tendo recebido duas nominações no *Game Critics Awards*, a de melhor jogo Ação/Aventura e melhor jogo para console, também foi considerado melhor jogo pela *Forbes*.

O exemplo nos mostra que a UDK/*Unreal 3* é um motor de jogo que atende não só as propostas acadêmicas, não só a proposta de estúdios pequenos como o do jogo referido anteriormente, mas também atende o requisito das produções milionárias das quais centenas ou milhares de pessoas trabalham, é um motor de jogo que atende as expectativas gráficas e estéticas para grandes produções, como podemos ver pelos *prints* abaixo:

Figura 18: Batmam Arkham Origins (2)

Figura 19: Batmam em Combate

O jogo se trata do que se chama de *prequel*[15] de *Batmam Arkham Asylum*, o primeiro jogo da série, mostra um Batmam mais jovem que combate o maior assassino de Gotham, Black Mask, dentre os vilões também estão incluídos o Coringa e Anarky.

2.3.3 – Bioshock Infinite

Figura 20: Bioshock Infinite (2013)

Bioshock Infinite (2013) é um jogo digital em 3D, em primeira pessoa, classificado pela ESRB com o gênero *First Person/Shooter* (isso é, jogo de tiro em primeira pessoa), é um jogo multiplataforma, disponível para PC, *Playstation 3*, *XBOX 360* e *Mac OS X*, lançado em 2013 pela *2K Games* e produzido pela *Irrational Games*, idealizado pelo *game designer* Ken Levine, produzido em *Unreal 3*.

Escolhemos esse jogo em especial por ele ser um jogo de tiro em primeira pessoa, lembrando que como dito anteriormente, inicialmente a *Unreal* tinha sido idealizada

[15] *Prequel* é o contrário de sequência, é na verdade o que vem antes, é uma versão que explica como tudo aconteceu.

para ser um motor de jogo voltado para a produção de jogos de tiro em primeira pessoa e por mais que atualmente não seja mais assim, o modo tiro em primeira pessoa já vem programado pronto na UDK/*Unreal 3*. Também o escolhemos por ser um jogo *Triple A* de grande expressão no mundo dos jogos, sendo de uma franquia famosa e conhecida por sua alta qualidade, tendo ganhado na E3 o prêmio de melhor jogo da feira em 2013. O jogo ganhou 85 prêmios e vendeu mais de 4 milhões de cópias[16].

Como podemos ver nas imagens abaixo, é um jogo de grande qualidade estética:

Figura 21: Cidade de Columbia, Cores e Saturação

[16] Informação disponível em http://en.wikipedia.org/wiki/BioShock_Infinite e http://pt.wikipedia.org/wiki/BioShock_Infinite Acessado em 10/2014;

Figura 22: Elizabeth, uso da Iluminação para Focar o que tem de ser Focado e Esconder o que tem de ser Escondido

Como podemos ver, o jogo tem ora uma iluminação clara que realça as cores que já são bastante saturadas e ora um ambiente mais escuro do qual a luz foca o que tem de ser focado e oculta o que tem de ser ocultado, esse também foi um dos motivos de termos escolhido esse jogo, afinal a *Unreal 3*/UDK são conhecidas pela ótima geração de luz e sombras.

Bioshock Infinite (2013) é um jogo de que não se conecta com a história do resto da franquia, pois além dos personagens não serem os mesmos, a história também não se passa *Rapture*, se passa em *Columbia*, uma cidade flutuante, em 1912, da qual o agente Booker DeWitt é enviado para procurar uma mulher chamada Elizabeth. Após encontrar Elizabeth, ambos se envolvem em um conflito de facções na cidade, entre os Nativistas que se esforçam para manter os americanos da cidade puros e a *Vox Populi*, pessoas comuns rebeldes com a situação. Durante o jogo descobre-se que Elizabeth é uma personagem chave no que ocorre na cidade e ela também abre fendas no espaço-tempo em *Columbia*.

A cidade de *Columbia* é uma cidade utópica, cheia de cores fortes, vibrantes e com muita saturação, seria uma cidade perfeita, porém apenas na superfície.

Capítulo 3: Tutorial

Aqui demostramos os aspectos práticos da pesquisa visando à construção de um ambiente navegável e introduzindo as potencialidades do desenvolvimento de *games* e metaversos com o motor de jogo UDK.

O material aqui apresentado é um conjunto introdutório das potencialidades do motor de jogo, visando oferecer ao leitor um material para que este possa ter por onde iniciar o desenvolvimento do seu projeto de forma didática.

3.1 - Modelagem 3D

Para iniciar um projeto de *game* 3D na UDK, ou em qualquer outro motor de jogo, a etapa prática mais primária e necessária é a modelagem 3D. Contudo motores de jogo disponibilizados por companhias para o desenvolvimento, possuem suas especificidades até na hora da modelagem 3D.

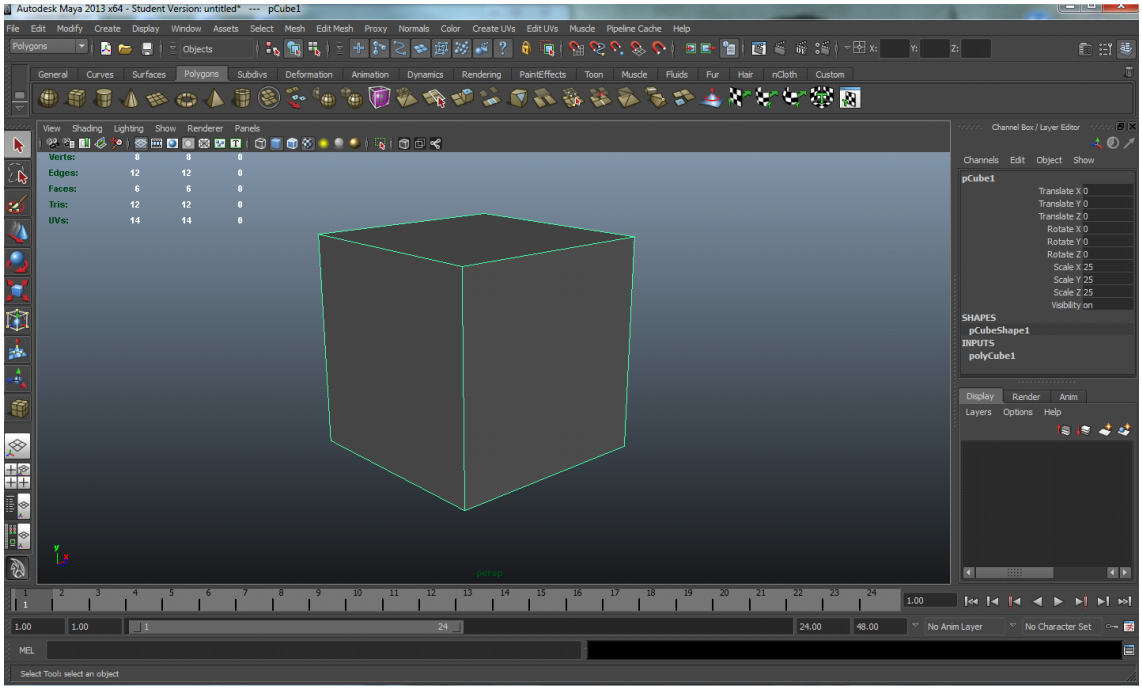

Figura 23: Cubo

A maior parte dos *assets* produzidos para jogos digitais começam por métodos tradicionais de modelagem, conhecidos como *poly by poly* e *box modeling*. No desenvolvimento atual vamos utilizar o processo de *box modeling*, uma forma de desenvolvimento que se inicia sempre partindo de uma forma compreendida dentre as três formas geométricas básicas, no caso: cubo, esfera ou cilindro.

Para a construção do elemento primário do nosso ambiente navegável começamos com um cubo.

As dimensões do cubo nessa circunstância primária podem variar, contudo é recomendável que o objeto possua uma escala superior a 10 (dez) no mínimo, escalam - se posições a partir de valores numéricos que podem ser editados no *channel box*, janela de edição localizada no canto direito do *Autodesk Maya 2013*. Ainda no *channel box*, é imprescindível que a posição do objeto esteja setada nos parâmetros zero. Isso é necessário porque o parâmetro zero no *Maya* é o centro da *grid* de orientação, e para o motor de jogo UDK, sua referência de *gizmo* para edição é o centro da *grid*.

Das várias ferramentas para modelagem 3D uma das que mais se faz uso é o comando *extrude.*

O *extrude* consiste em modificar uma face do polígono permitindo uma extensão ou retração da face, utilizamos essa ferramenta para dar origem ao nosso objeto.

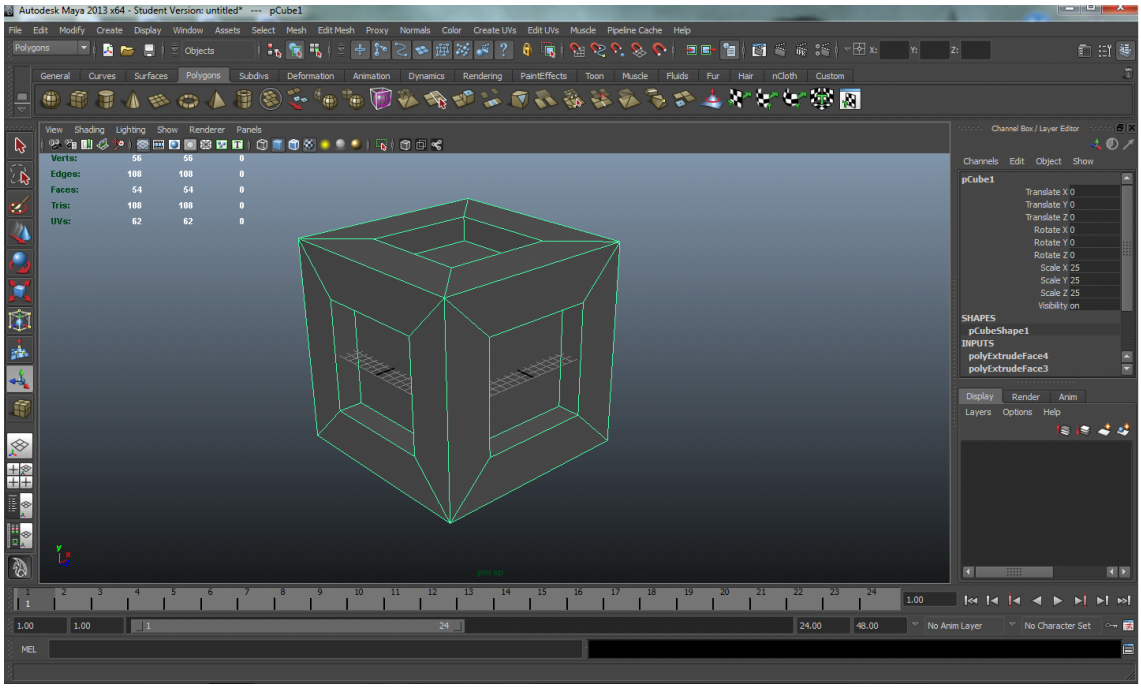

Figura 24: Extrude

3.2 - Mapeamento de Coordenadas UV

A modelagem 3D para jogos digitais difere da modelagem 3D para o cinema, a

modelagem para jogos necessita de truques para economizar no processamento, sendo assim é comum deixar uma ilusão de detalhamentos.

Para se criar essa ilusão utilizamos uma imagem criada por composição através de desenhos e fotografias, essa imagem é chamada de textura (como explicitamos no capítulo 1 da presente pesquisa). A aplicação de uma textura em um objeto 3D é desenvolvida a partir de um processo chamado mapeamento UV, que consiste segundo Rabin (2013, p. 691) em utilizar um sistema de coordenadas que estabeleça uma relação entre a imagem 2D e o objeto 3D. Essa relação parametriza o polígono e aplica nas faces agora desmontadas em um plano bidimensional cartesiano, uma projeção da imagem escolhida para ocupar aquele determinado espaço, no caso uma face do polígono.

Visando texturizar nosso objeto, vamos então criar esse mapeamento resultando na imagem abaixo:

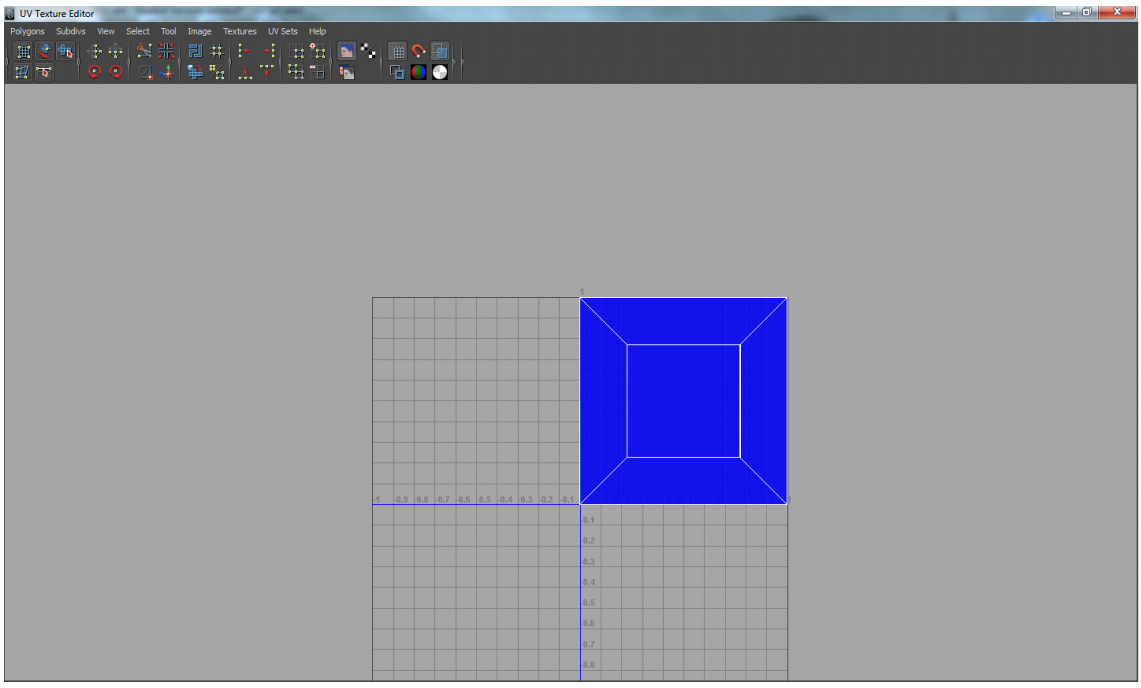

Figura 25: Mapeamento (1)

O UV *texture editor* do Maya é a ferramenta responsável por originar as projeções de textura no objeto 3D. Através de um mapeamento planificado nos eixo x, y e z obtemos

esse resultado.

Com esse mapeamento é possível construir uma textura própria para o objeto. Na modelagem 3D tradicional ou para cinema, a textura não precisa necessariamente ser condicionada ao objeto, o mesmo ocorre para o desenvolvimento de jogos digitais. Contudo o desenvolvimento de uma textura condicionada ao objeto é uma boa forma de ministrar o peso e a qualidade do comportamento da textura dentro do motor de jogo. Para criar essa textura condicionada utilizamos o mapeamento UV desenvolvido para aplicação e utilizado como um guia para produção da textura. Nesse caso o comportamento do objeto nos permite uma sobreposição.

Com sobreposição é possível ganhar em resolução, caso a textura não seja condicionada ao objeto, mas sim uma textura de repetição (*Tiled*) a sobreposição também tem um comportamento muito satisfatório.

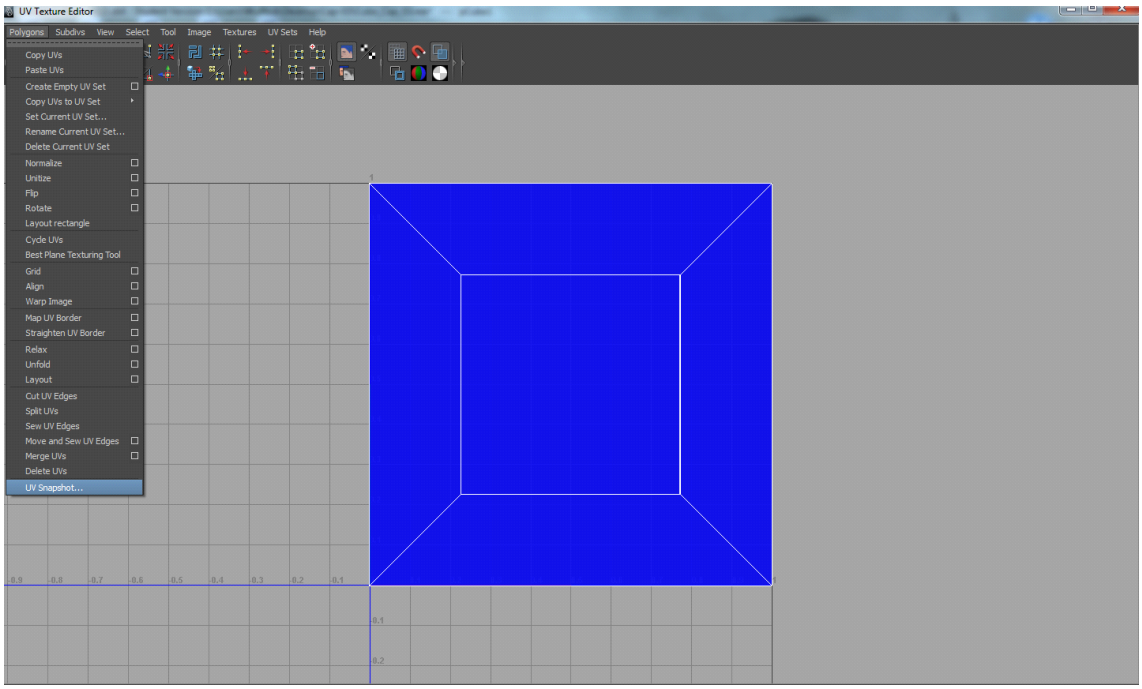

Figura 26: Mapeamento (2)

Utilizando a ferramenta *UV SnapShoot* do modelador no caso *Maya*, vamos tirar uma foto da nossa UV, gerando uma imagem da nossa UV. Essa imagem será nosso guia na produção da textura, pois sobre ela que "pintaremos" a textura em um *software* de

edição 2D como o *Photoshop* por exemplo.

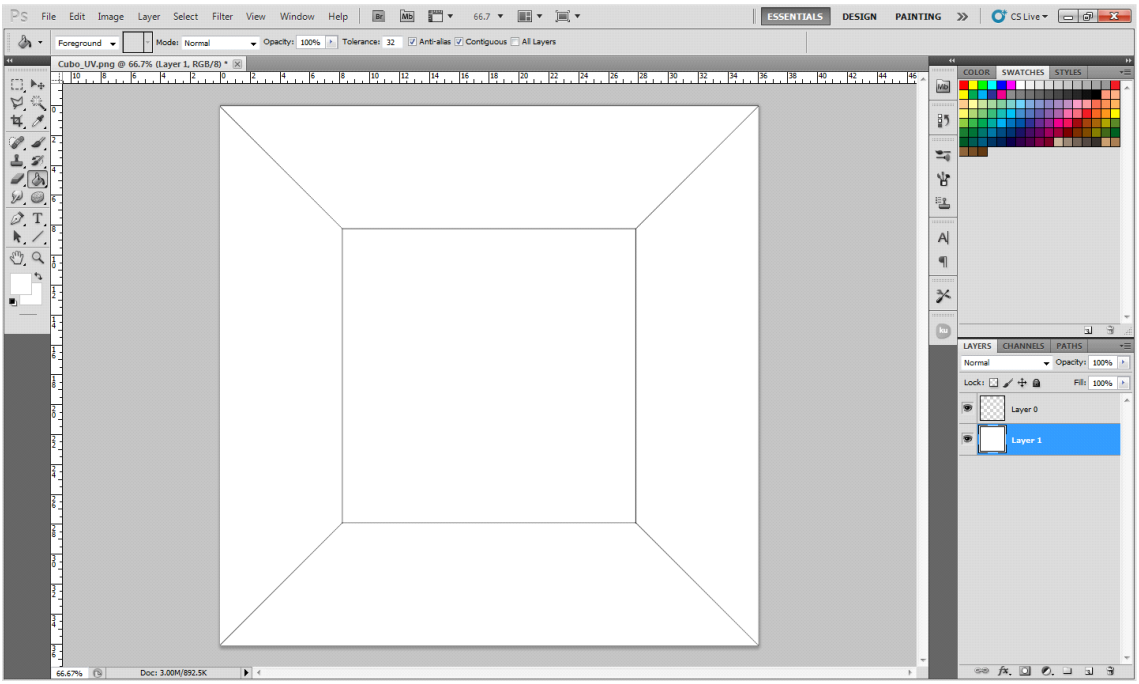

Figura 27: Snapshot no Photoshop para produção da textura

Em um editor de imagem, no caso o utilizado aqui é o *Photoshop*, abrimos a imagem da nossa UV, ao inserirmos teremos um guia para produção da textura.

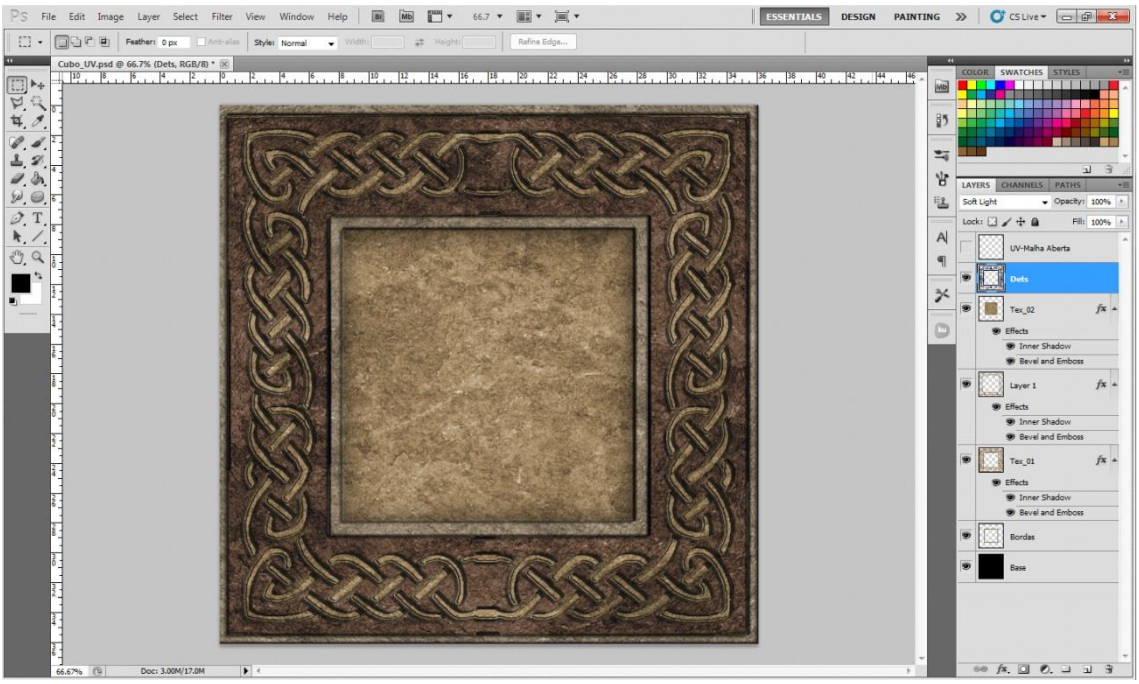

Figura 28: Produção da Textura

Utilizando alguns *brushs* do *Photoshop* e uma textura de base, damos origem a nossa textura condicionada ao objeto.

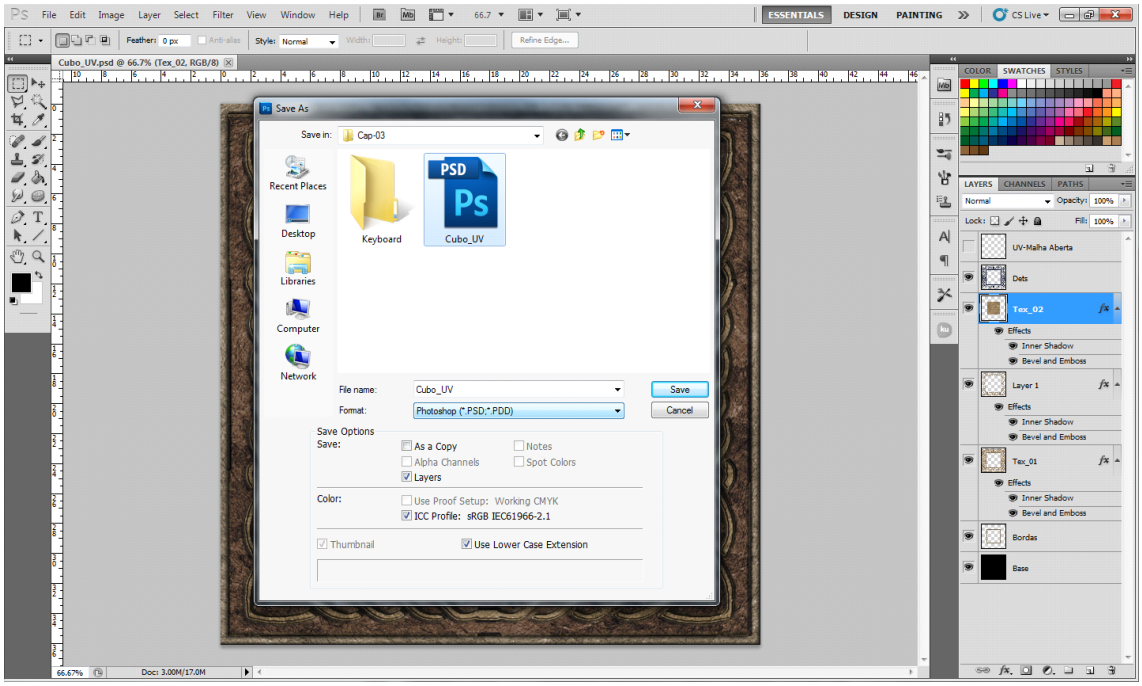

Figura 29: Salvando a Textura

Damos origem a um arquivo PSD, é importante se atentar a extensão da textura, pois motores de jogos geralmente possuem restrições com algumas extensões de imagem. No caso da UDK, as mais recomendadas são PSD, TGA e PNG. A escolha de PSD nesse caso é para facilitar a exportação do nosso objeto para o motor de jogo.

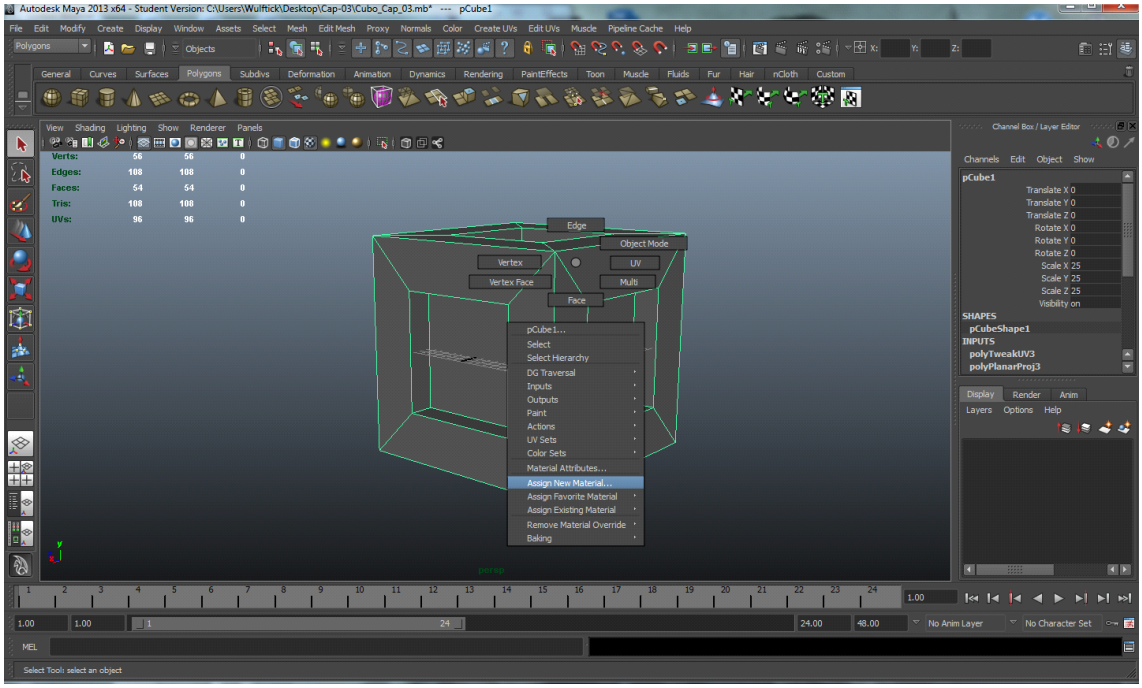

Figura 30: De volta ao modelador: Colocando a Textura

De volta ao modelador vamos aplicar um novo material ao nosso objeto, agora aplicando a textura nesse mesmo material.

Clicamos como botão direito *mouse* e escolhemos a opção *Assign New Material*, como na imagem acima.

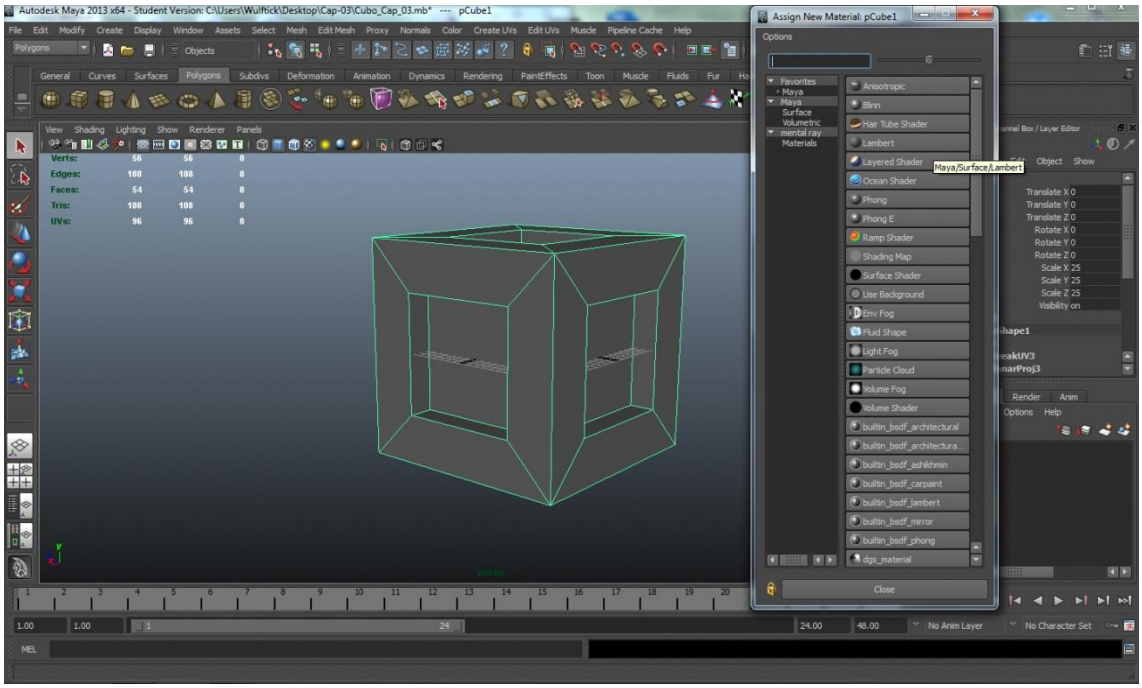

Figura 31: De volta ao Modelador: Colocando a Textura (2)

De volta ao modelador vamos aplicar um novo material ao nosso objeto, agora aplicando a textura nesse mesmo material.

Vale lembrar que os materiais nos modeladores 3D são responsáveis por várias aplicações de *shaders* e efeitos no objeto, contudo são raros os motores de jogo que podem trazer os atributos de *shaders* do material de um modelador 3D para aplicação dentro do mesmo.

Nesse caso, é recomendável utilizar um editor de material paralelo como *Substance* ou realizar a reconstrução do *shader* no motor de jogo, que é no caso a nossa forma de produção para este objeto.

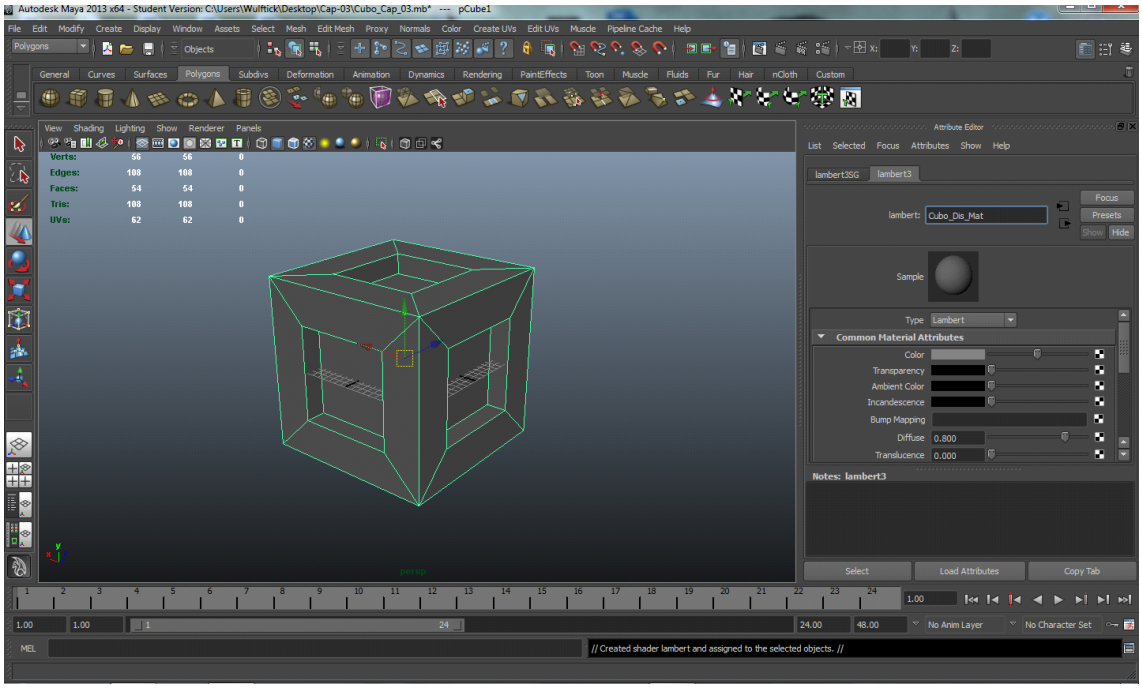

Figura 32: Renomeando Material

Aqui nós renomeamos o material, notem que ao invés de usar o espaço, utilizamos *underline* (_), pois a UDK não aceita o caractere espaço na composição de nomes. Essa informação é importante porque nomes de áudio, modelos tridimensionais e texturas não podem conter espaços. A aplicação do material aqui e a atenção ao seu nome se dá porque essas informações vão constar no arquivo FBX, que posteriormente se tornará o material do objeto no motor de jogo.

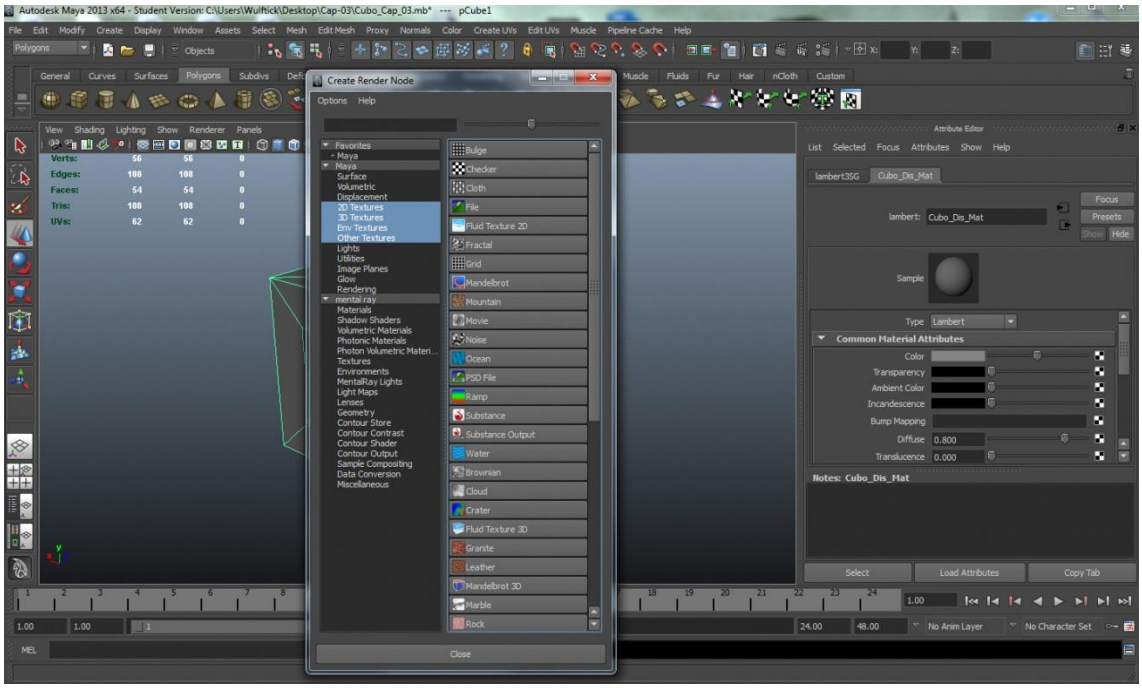

Figura 33: Setando a Textura

Agora vamos setar a textura, clicando no quadradinho em xadrez na opção *color* localizada no canto direito da tela, ao abrir a tela *create render node*, iremos selecionar a opção *file*, nos permitindo setar nossa textura.

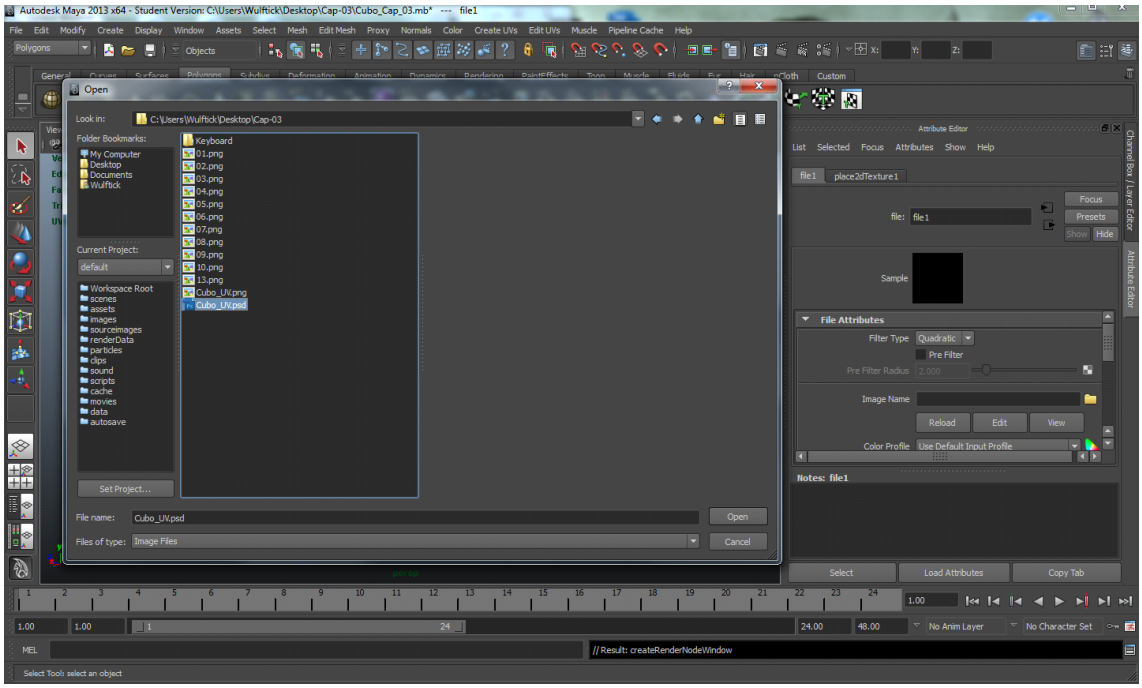

Figura 34: Selecionando a Textura

Agora vamos selecionar a imagem que será a nossa textura. Após clicar em *file* na janela anterior, teremos uma janela nova no canto direito com uma pasta na opção *image name*. Clicando no ícone pasta, seremos capazes de selecionar a imagem 2D (Textura) que será aplicada no nosso objeto 3D. Nesse momento vale a pena lembrar que a UDK é bastante seletiva com os formatos aceitos, nesse caso escolhemos PSD.

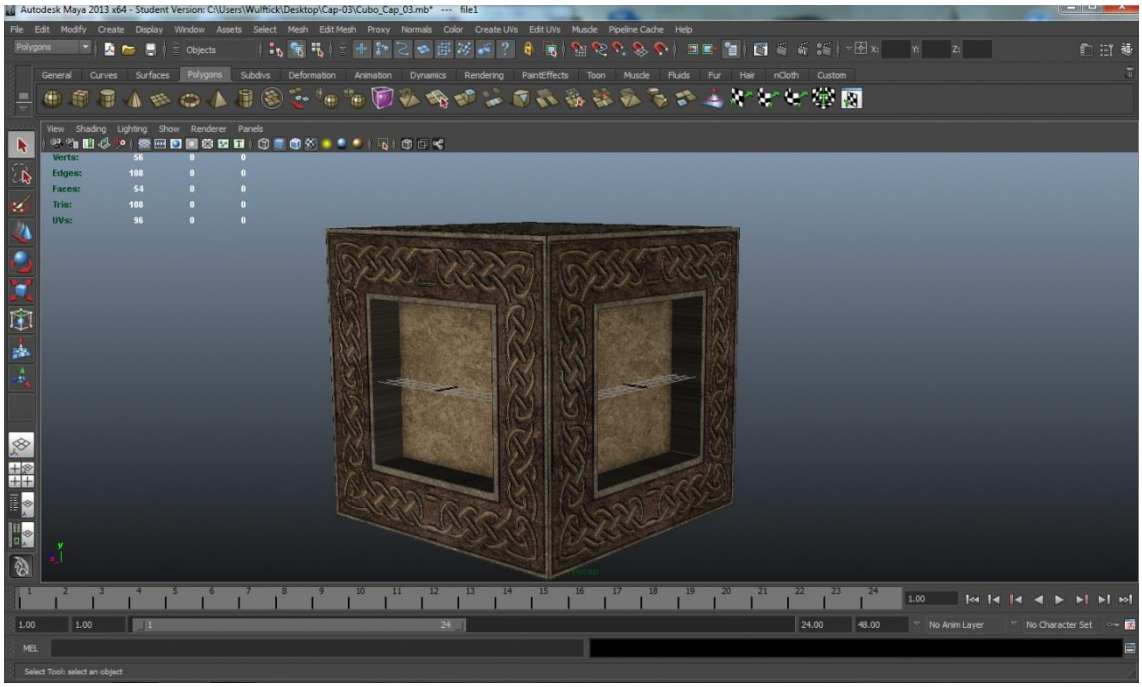

Figura 35: Objeto 3D Texturizado

Aqui podemos observar o objeto 3D agora com material e texturas devidamente aplicadas, quase pronto para ser enviado para UDK.

Contudo ainda precisamos pensar em um aspecto, um aspecto específico da UDK, o *lightmap*.

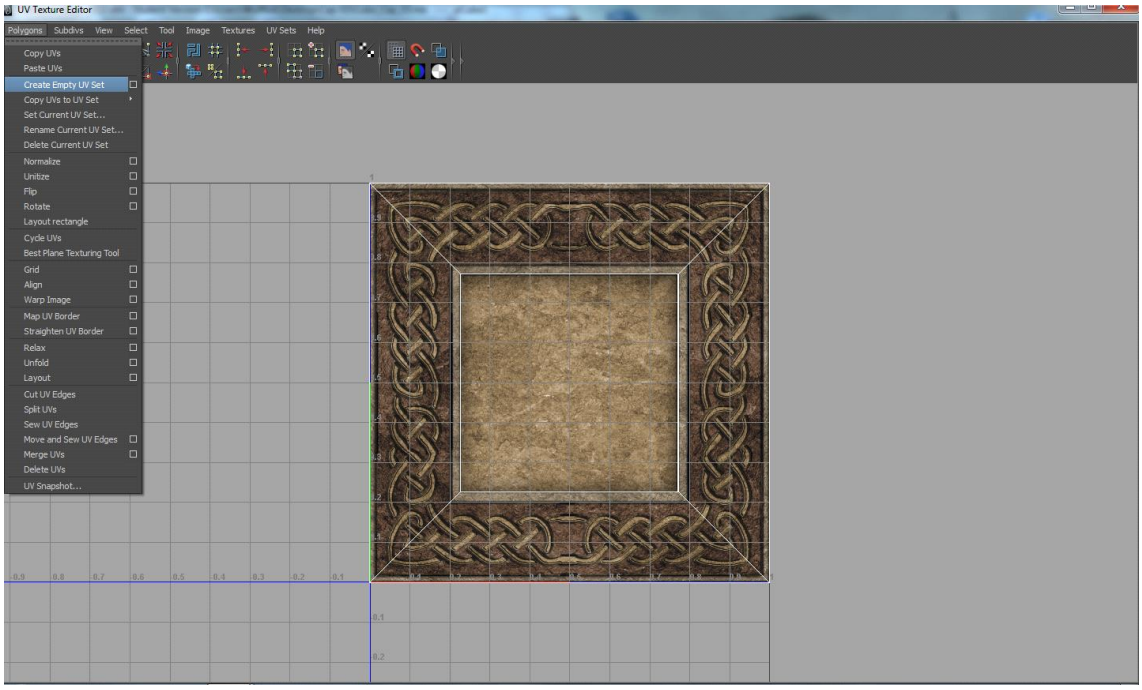

Figura 36: Lightmap

Aqui nós abrimos o *UV texture editor* e vamos criar o novo UV set (que será o *lightmap*). A UDK tem um sistema de iluminação próprio e a mesma é muito conhecida por ele, no entanto se o *lightmap* (mapa de iluminação, que indica como a iluminação do motor de jogo refletirá no objeto) não é criado, quando a iluminação gerada pela UDK, bate no objeto 3D, o mesmo fica com manchas pretas e sombras e com o *lightmap* a iluminação bate normalmente no objeto, gerando sombras coerentes.

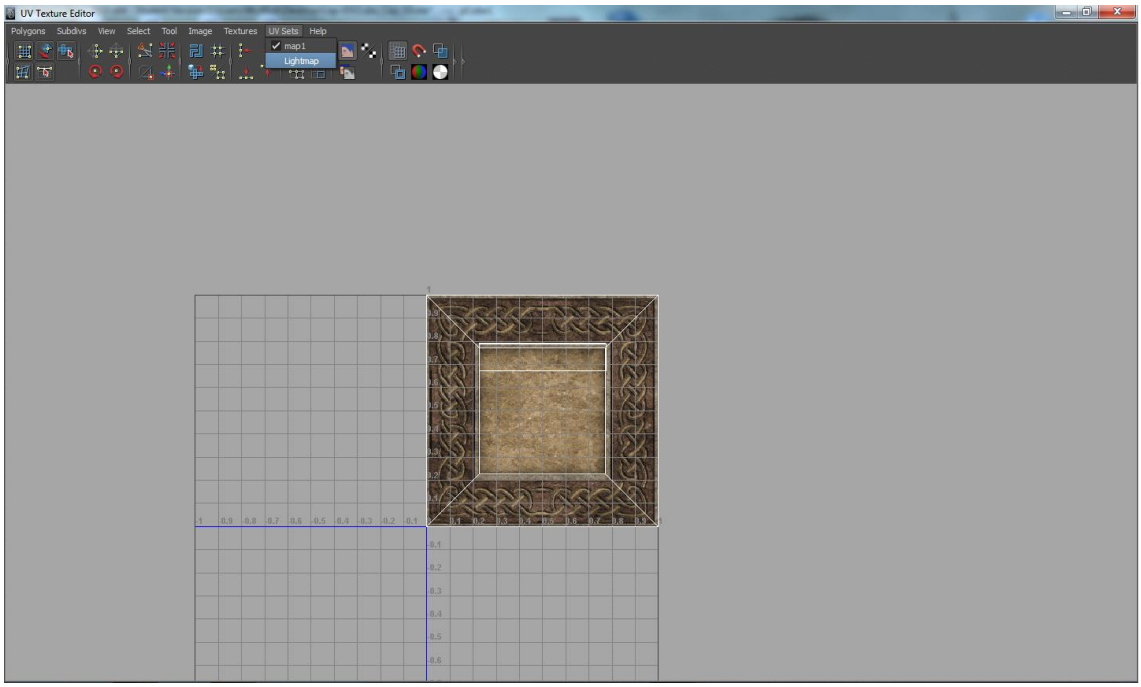

Figura 37: Lightmap (2)

Nessa janela iremos selecionar o *UV set* recém criado para desenvolver o *lightmap*.

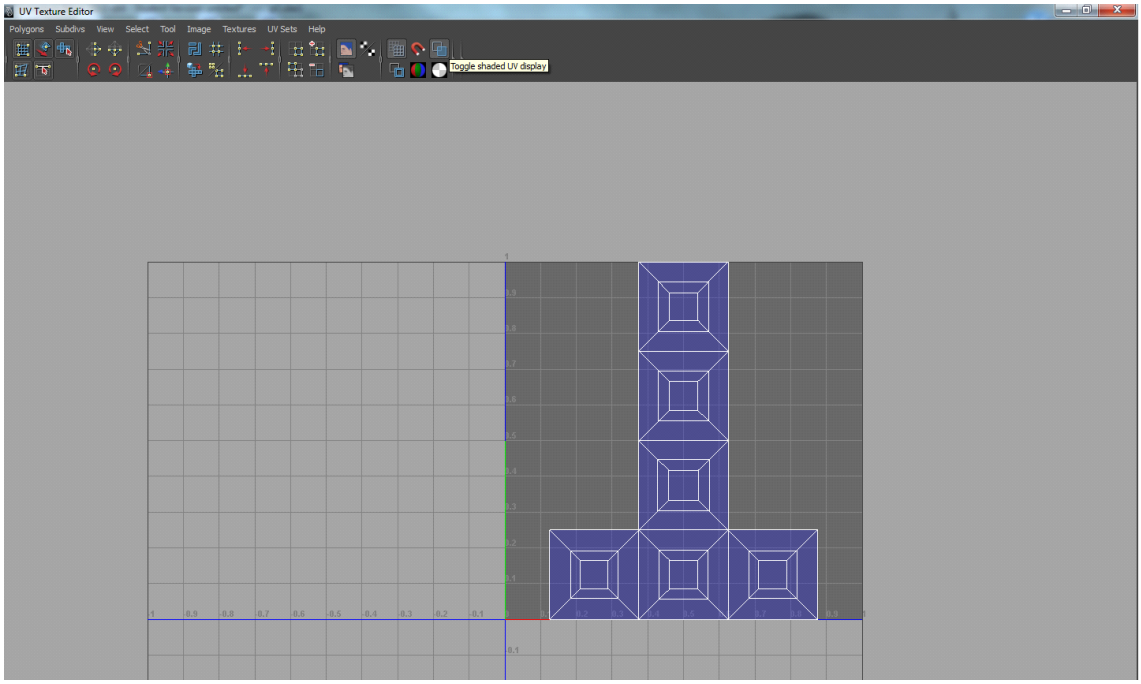

Figura 38: Parametrização

Aqui que o *lightmap* se dá, uma realização da parametrização do objeto do qual

nenhuma face se sobrepõe, como vemos na imagem acima.

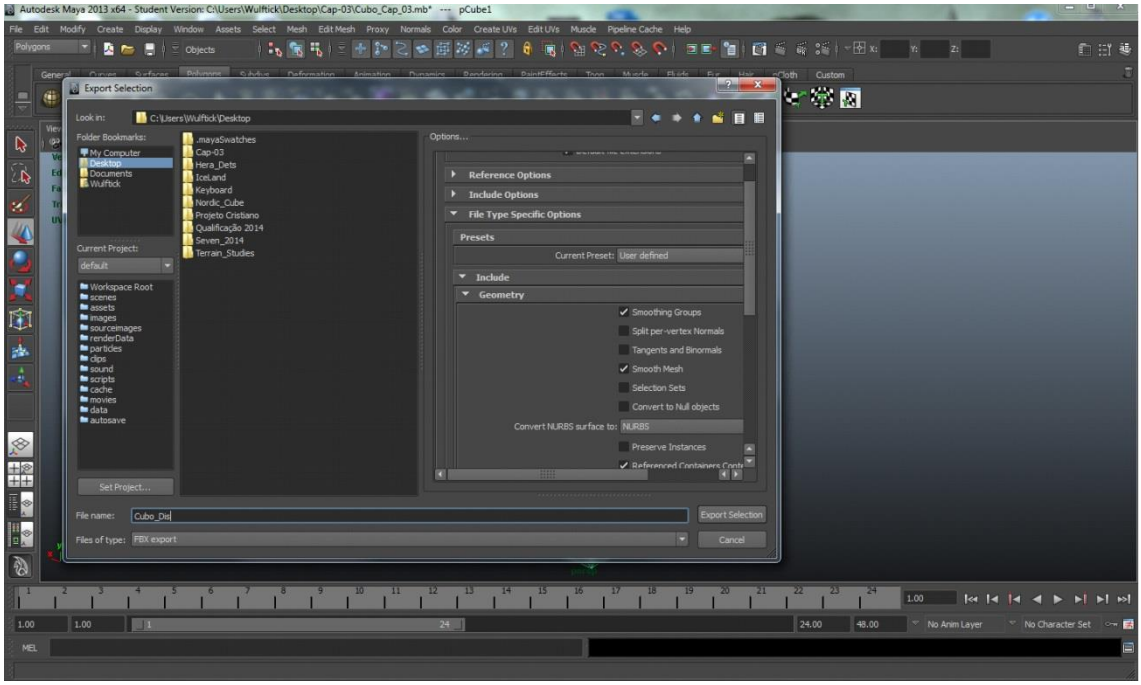

Figura 39: Exportação em FBX

Aqui estamos realizando a exportação do arquivo em formato FBX para a UDK. Clicando em *file*, *export* e em *file type* selecionando FBX *export*.

A questão técnica do *lightmap* aqui explicitada nos evoca a questão do *lightdesign*, que nada mais é do que o conceito de arquitetar a iluminação em jogos digitais criar texturas, sombreamentos, matizes de cores, iluminar, mostrar ou esconder através da iluminação. Esse conceito foi trabalhado por nós em um artigo intitulado: *A Influência da Pintura Barroca na Produção do Light Design nos Jogos Digitais: Estudo de Caso Dead Space 2* (Marques; Petry, 2014), o artigo foi apresentado no SBGames em novembro de 2014, que ocorreu em Porto Alegre – Rio Grande do Sul.

O artigo visa a investigação das influências da pintura barroca na produção do *lightdesign* de jogos digitais, utilizando como caso modelar o jogo digital *Dead Space 2*. O artigo conta com citações retiradas do diário de produção do jogo que explicitam todo o processo de *lightdesign* em jogos.

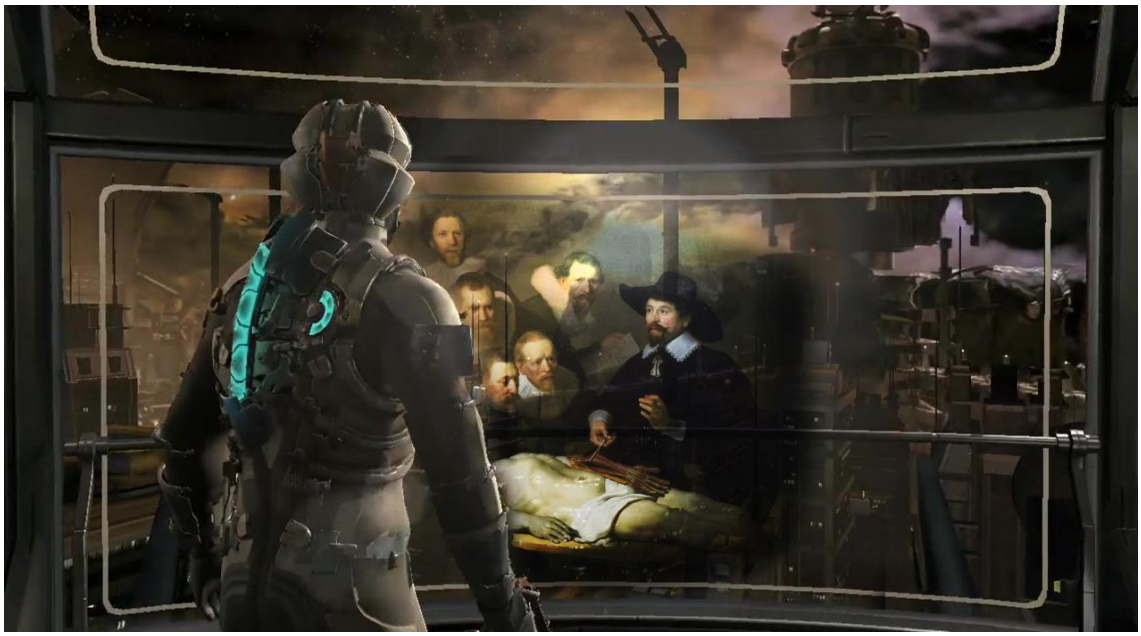

Figura 40: Luz, Dead Space e Rembrandt

> "Nos vídeos sobre o desenvolvimento de *Dead Space 2*, o diário de desenvolvimento aborda a construção da atmosfera de horror do jogo, Iam Milham diretor de arte do jogo, aborda como ocorreu a construção da identidade visual do jogo, mais especificamente a importância da iluminação para o desenvolvimento da atmosfera que a equipe da Visceral Studio buscava oferecer para o jogador. Buscando nos grandes mestres do passado, mais especificamente em pintores holandeses do barroco, a equipe de Ian buscou entender como os grandes mestres distribuíam seus pontos luminosos e o comportamento dos mesmos nos elementos da pintura. Estudando Rembrandt, partindo de suas famosas pinturas *Aula de anatomia do Doutor Tulp* e *Ronda Noturna*, o vídeo mostra como a influência das distribuições de luz nos quadros contribuíram para o desenvolvimento de *Dead Space 2*. Fica evidente uma distribuição inteligente dos elementos narrativos/imagéticos e de suas modificações controladas quando os vários tipos de iluminação como Pointlight ou Spotlight entram em contato com tais elementos." (Marques; Petry, 2014, p. 4).

Como podemos ver nas imagens abaixo, a iluminação da um enfoque especial nos objetos dos quais deve - se focar atenção no jogo.

> "Fenômeno esse análogo ao que explicamos anteriormente de que na pintura barroca, especialmente na rebrandtiana, a luz direciona o foco de quem está olhando, para locais, personagens ou objetos de maior importância, focando o que tem de ser focado, podemos ver nitidamente esse processo nas imagens que se seguem e não só isso, mas fortemente a questão do claro-escuro" (Marques; Petry, 2014, p. 4).

Figura 41: Render de Dead Space 2: Pointlight

Figura 42: Render de Dead Space 2 Spotlight

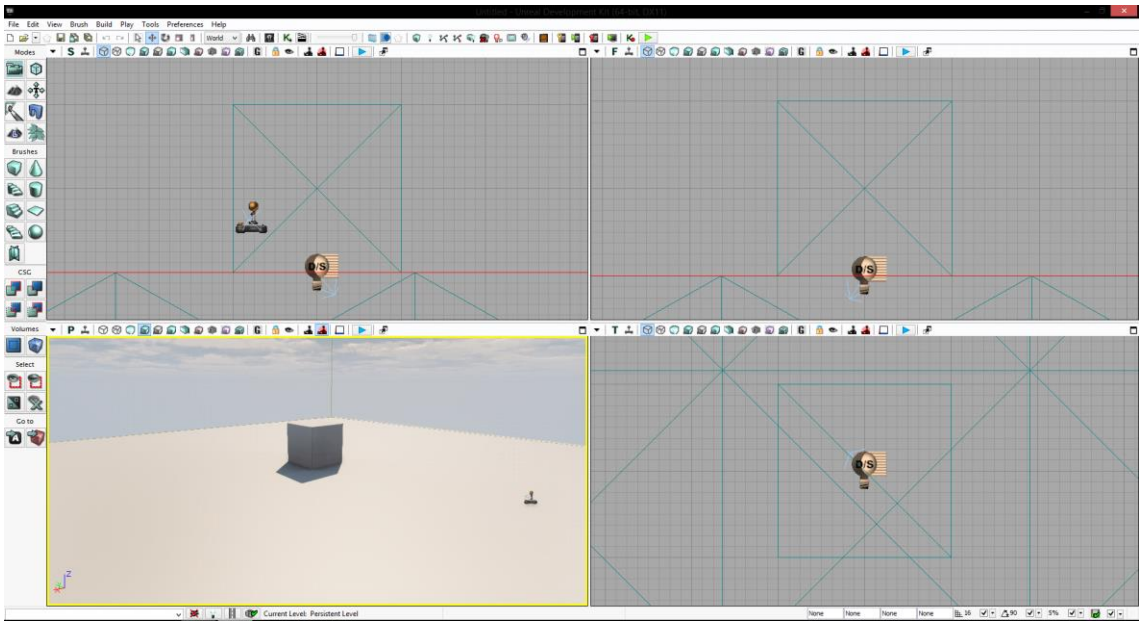

Figura 43: UDK Aberta, o Level Editor

Depois de termos exportado nosso primeiro objeto com material aplicado e devidamente texturizado, vamos passar para o desenvolvimento na UDK.

Nessa imagem, vemos o Editor da UDK, local onde o *level design* vai ocorrer aonde o jogo é "montado" (explicamos o termo e conceito *level design* no capítulo 1 da presente pesquisa). Na janela observamos 4 (quatro) tipos de visões semelhante ao modelador 3D.

3.3 – Content Browser e Packages

Para começar a introduzir o nosso conteúdo dentro do motor de jogo, vamos utilizar o gerenciador de *assets* da UDK chamado *content browser*, localizado na barra superior do editor.

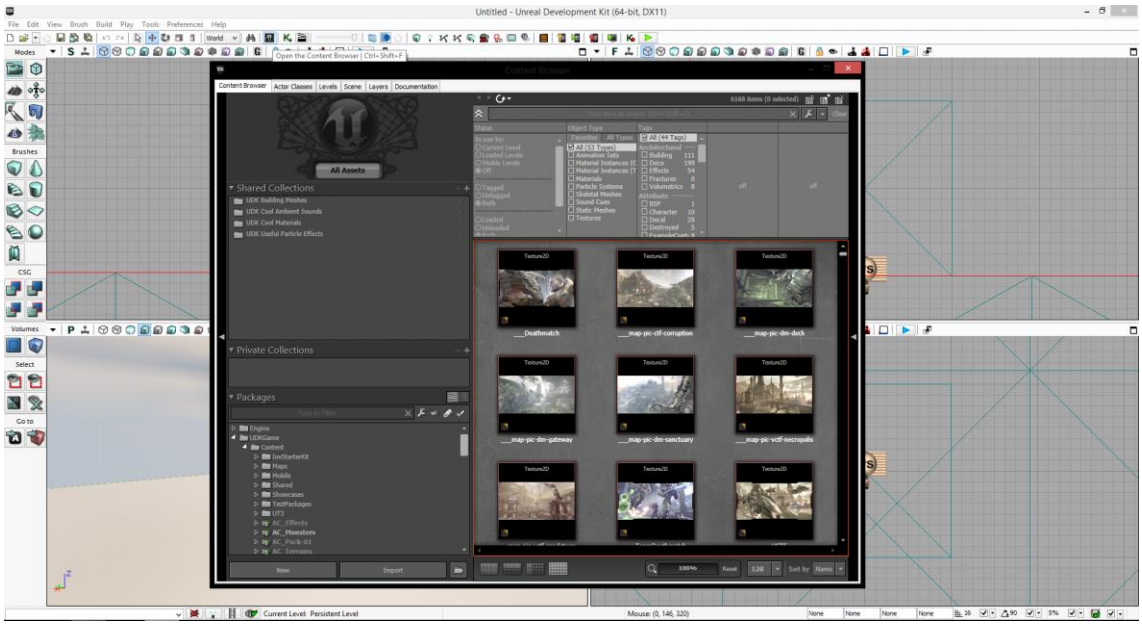

Figura 44: Content Browser

Daí clicarmos no ícone indicado, teremos acesso ao *Content Browser*, nessa janela somos capazes de gerenciar todos os *assets* disponíveis na UDK para desenvolvimento. Vale lembrar que pelas normas de utilização não é permitida a utilização comercial ou não comercial de *Skeletal Meshs* (isto é, objetos 3D disponíveis dentro da UDK que foram utilizados por eles no *Unreal Tournament*, jogo demonstrativo que é lançado quase junto com cada versão de *Unreal*, estando esses objetos disponíveis apenas para teste e para ver) isso se deve pela presença deles em outros projetos como *Gears of War* e *Unreal Tournament 3*.

No *content browser* além de gerenciar os *assets* disponíveis, é possível colocar e organizar novos *assets* para seu projeto. Assim vamos trazer nosso cubo para o motor de jogo.

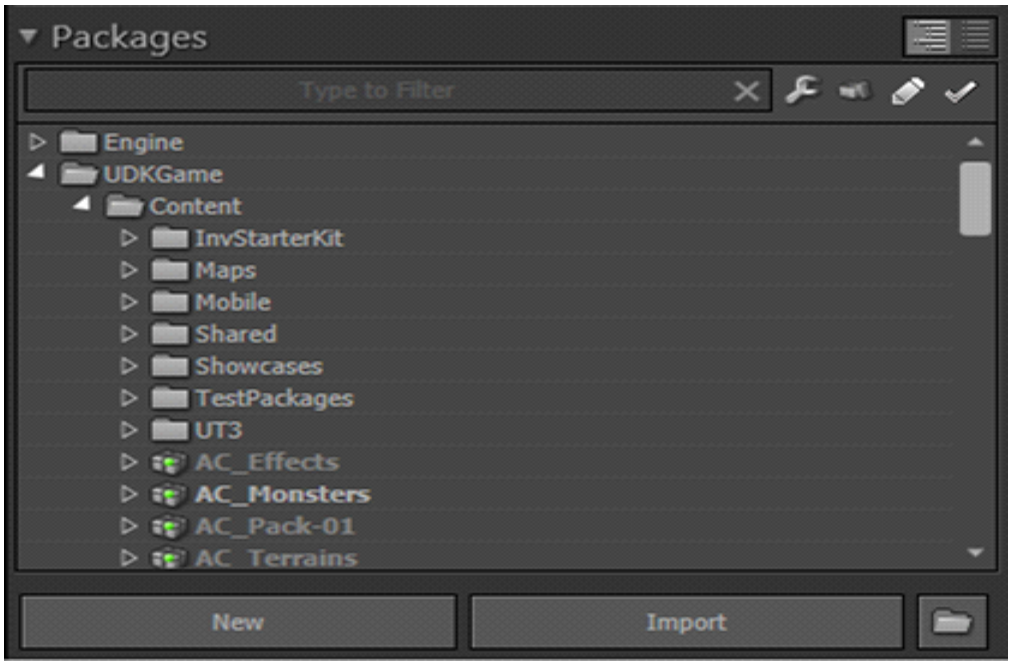

Figura 45: Packages

Notem que no canto inferior esquerdo existe um conjunto de pastas na aba *Packages*. Estas pastas são os locais onde se armazenam os conteúdos que formam o seu projeto, no caso modelos 3D, materiais e texturas. Nos *packages* também armazenam sons e arquivos SWF[17] de composição de HUDSs e GUIs[18].

Para Trazermos nosso objeto para dentro da UDK, iremos criar um *package* para o nosso projeto. O comportamento de um *package* é semelhante ao de uma pasta, nesta pasta você pode organizar outras subpastas que comportam um montante de arquivos do mesmo tipo.

Para criar nosso *package* novo vamos clicar no botão *new* ao lado de *import*.

[17] Shockwave Flash, trata-se de um formato de arquivo da Adobe, do qual é utilizado para inserir conteúdo multimídia, gráficas entre outras, tendo sido largamente utilizado para web e utilizado para jogos para fazer a GUI, a HUD entre outros.

[18] A HUD Head Up Display é uma barra que aparece normalmente em alguns jogos na parte superior da tela, indicando atributos como quantidade de magia, vida, entre outros. A GUI, Graphical User Interface, é a interface gráfica utilizada pelo usuário.

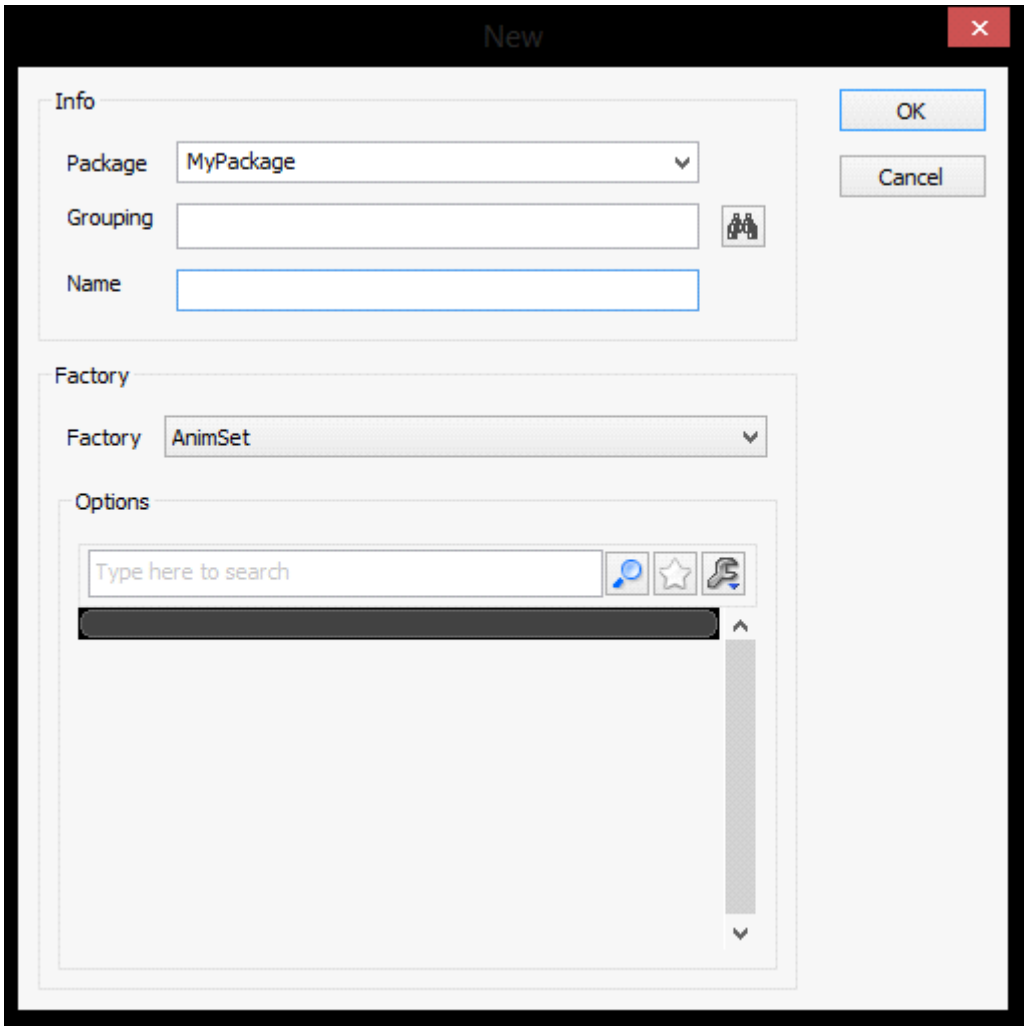

Figura 46: Setando o Nome do Package

No local onde está escrito *MyPackage*, é onde vamos setar o nome do nosso *package*, vale lembrar que uma vez setado, não será possível alterar seu nome nem mesmo no arquivo nativo. Para este momento vamos setar o nome do nosso package, para Puc_Cubo.

Ao criarmos nosso *package* novo ele automaticamente irá setar alguns *assets* da biblioteca para que este não fique vazio e tenha algum conteúdo, *packages* vazios não são lidos pela UDK, para evitar isso na criação ela seta um *asset* que pode ser deletado após você aplicar seu conteúdo no *package*, aqui no caso foi setado o *Anim Set*, que logo a seguir iremos deletar.

Sendo assim vamos importar nosso cubo criado previamente para o nosso *package*. Clicando com o botão direito em cima do Package.

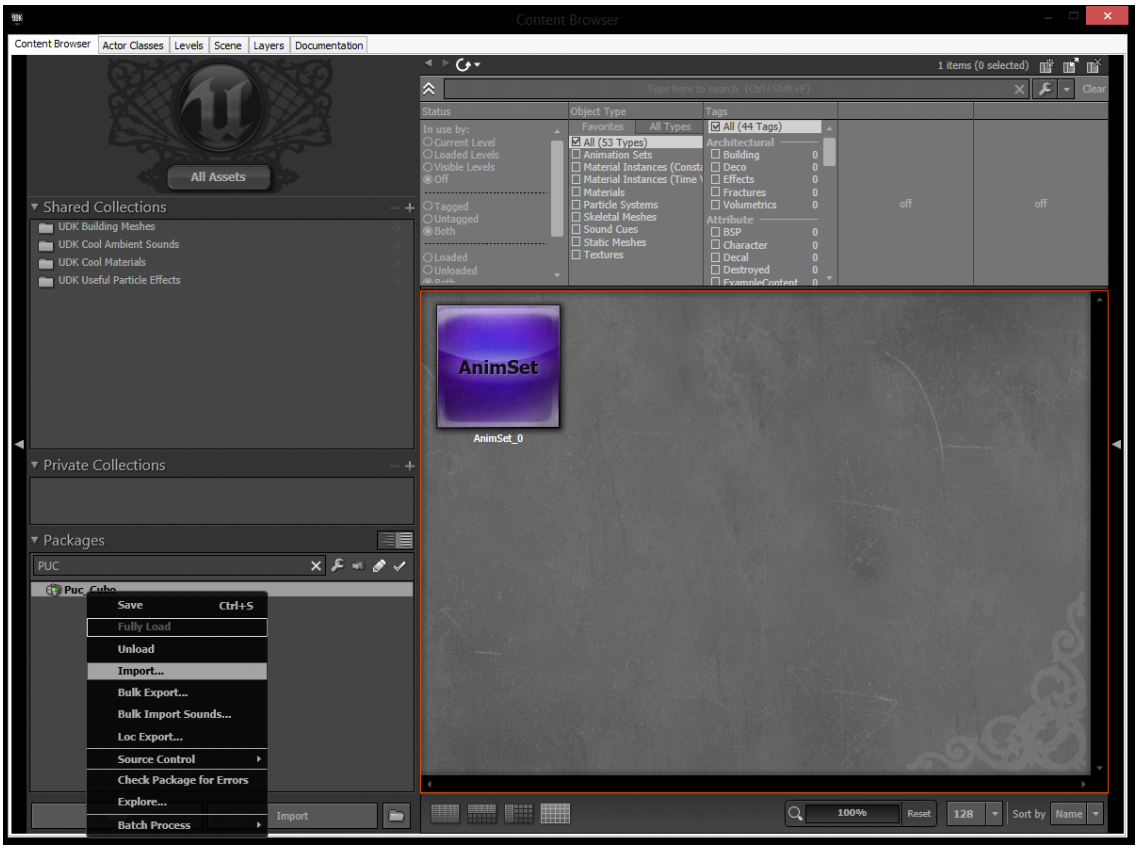

Figura 47: Package

E vamos selecionar nosso cubo, os *assets* desejados, no local em que ele foi armazenado, lembrando que ele deve estar exportado no formato FBX.

Ao selecionarmos o mesmo teremos acesso a esta janela:

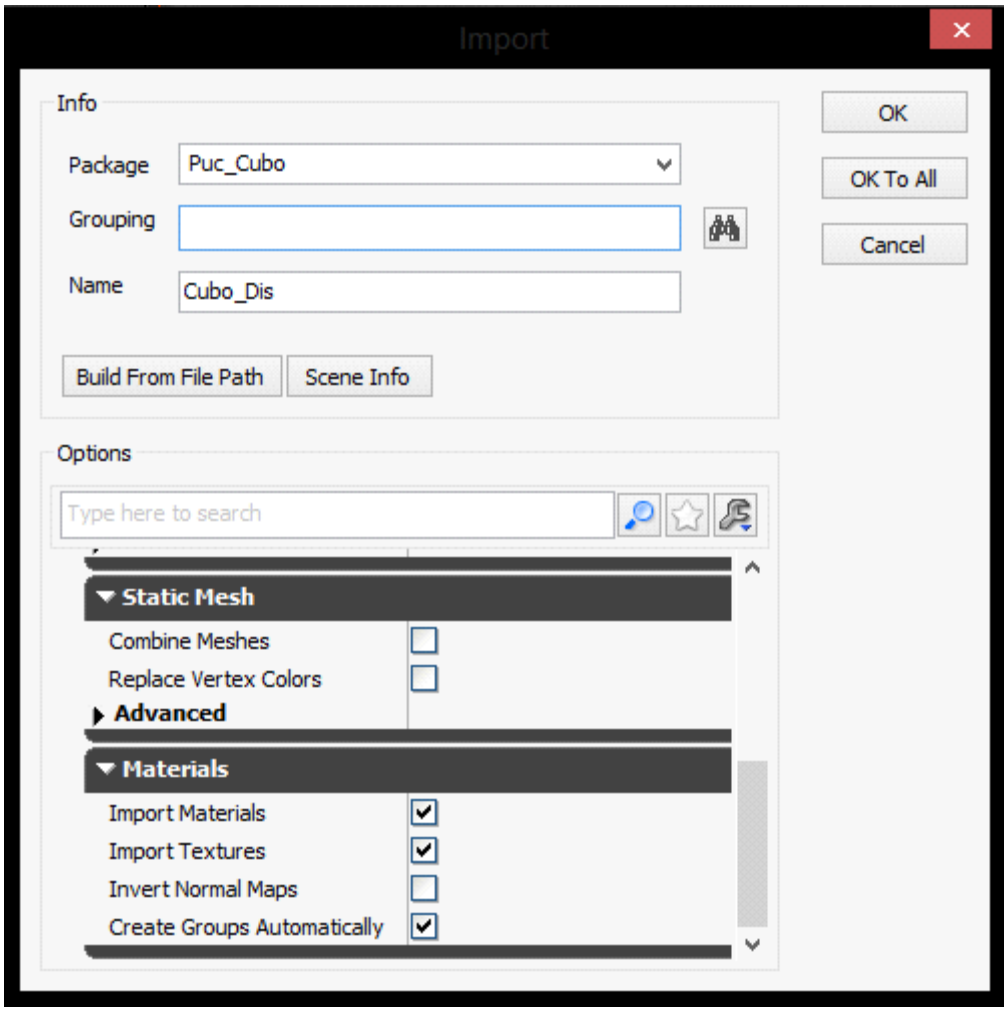

Figura 48: Package (2)

A opção *Grouping* é o local onde podemos manter uma organização sobre os elementos do *package*, é nesta opção que setamos pastas para armazenar nossos *assets* de forma organizada, como por exemplo, uma pasta chamada *Meshs* para armazenar nossos *assets* de modelos 3D.

Nativamente com a opção *import textures* e *import material* ativas, pastas com o nome respectivo, *Materials* e *Textures* (materiais e texturas) serão criadas, armazenando justamente os *assets* deste tipo, caso esta opção não esteja ativa, para manter uma organização sobre os *assets* do seu projeto e valido ativá-las.

Após estas etapas teremos este resultado:

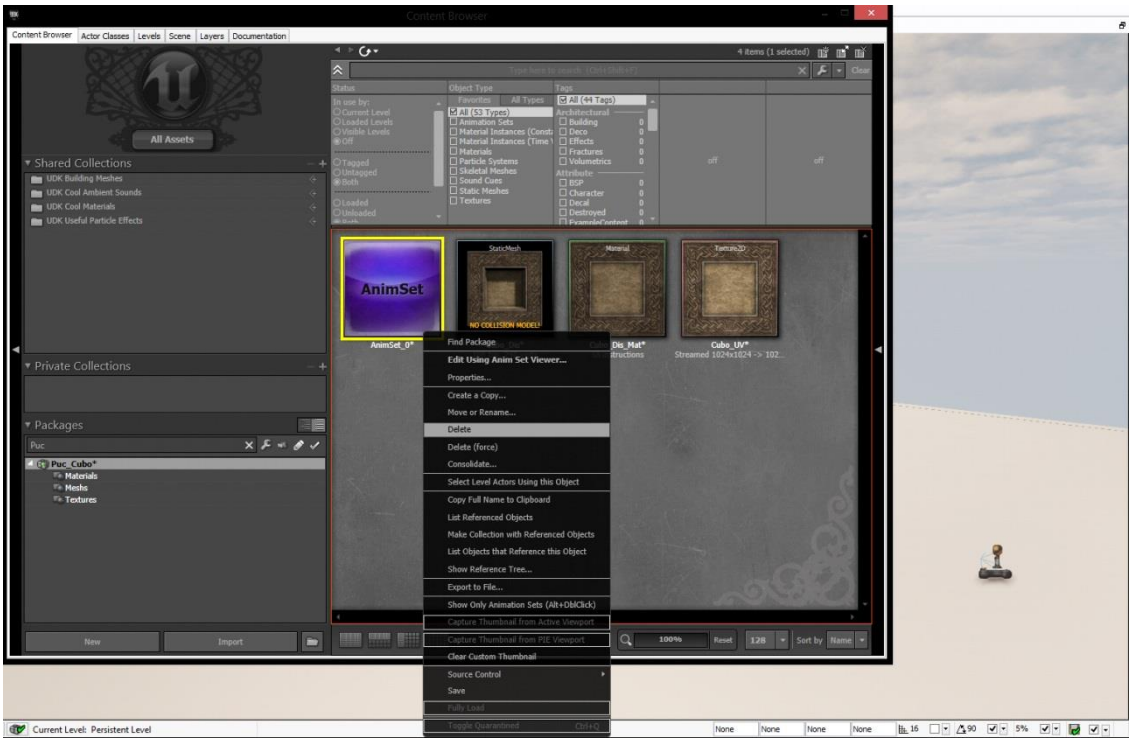

Figura 49: Deletando Anim Set

Lembrando que ainda temos esse *anim set* que foi colocado pela UDK para manter nosso novo *package* ativo, agora que ele possui conteúdo ele (o *anim set*) não é mais necessário, sendo assim vamos clicar com o direito e deletar esse elemento do nosso *package*.

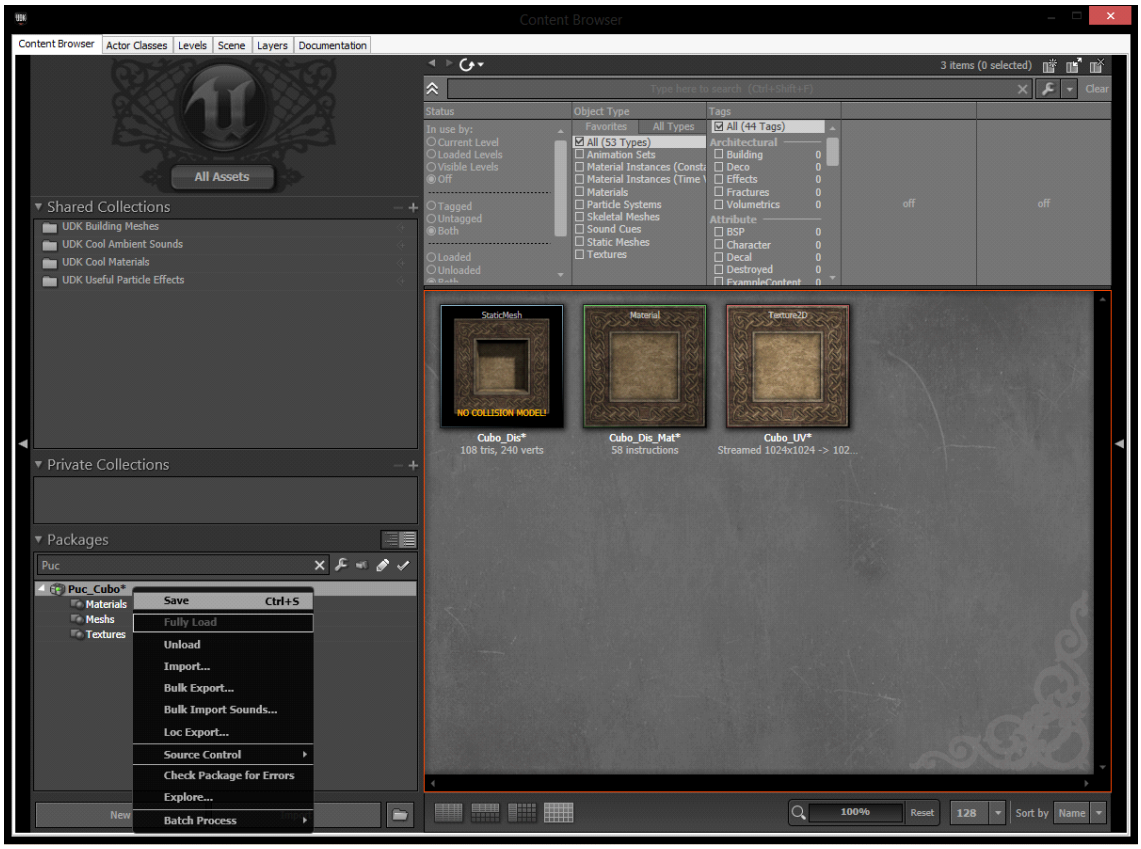

Figura 50: Salvando Package

Após deletarmos, podemos salvar nosso *package*, clicando com o botão direito em cima dele e selecionando salve. Sempre que um elemento estiver com a indicação de um asterisco no nome, quer dizer que este *package* ou elemento precisa ser salvo.

Sempre que criamos um *package* novo, quando vamos salvar pela primeira, vez a UDK vai abrir uma janela periférica perguntando o endereço em que você deseja salvar o novo *package*, é altamente recomendável que você mantenha o seu *package* que está no formato UPK junto com os outros *packages* nativos da UDK na pasta dentro da pasta *Content*.

3.4 – Static Meshs e Collision

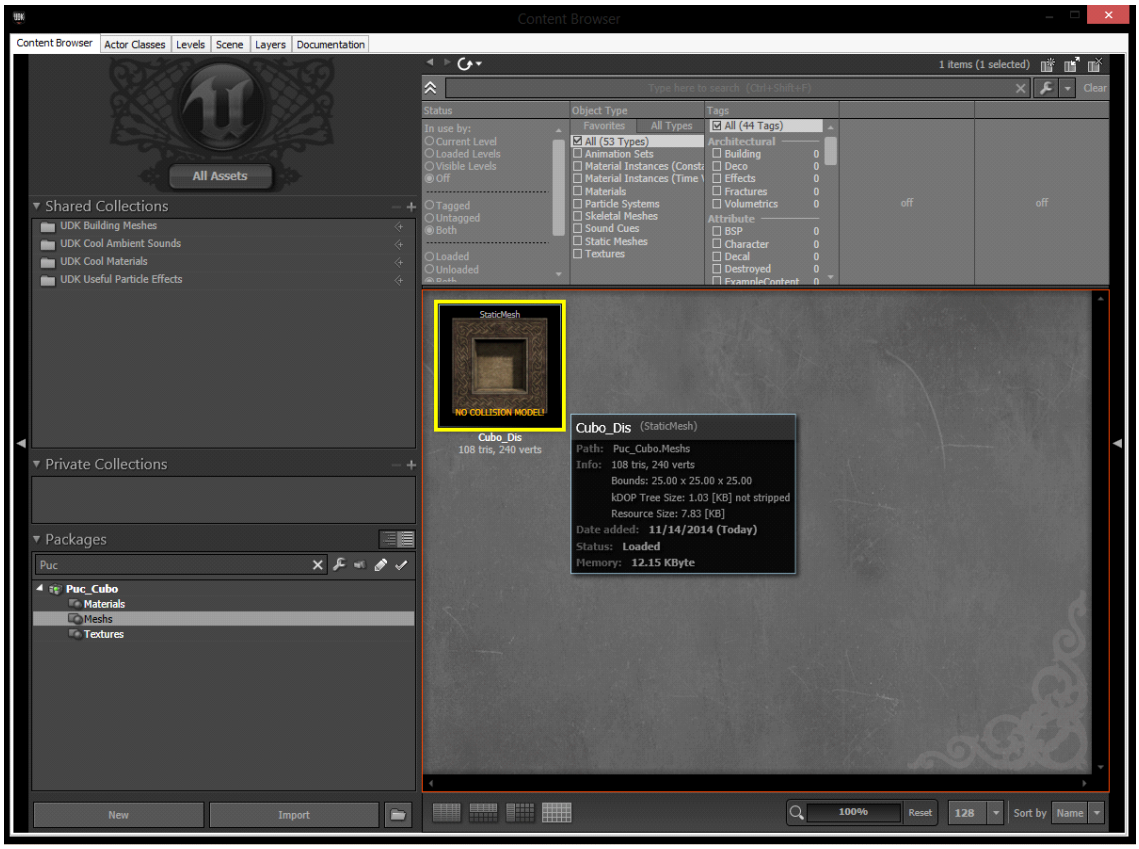

Figura 51: Static Mesh

Dentro da UDK elementos 3D são divididos em algumas categorias, para esta introdução vamos trabalhar com alguns deles. O mais presente no desenvolvimento de um projeto é a *static mesh*.

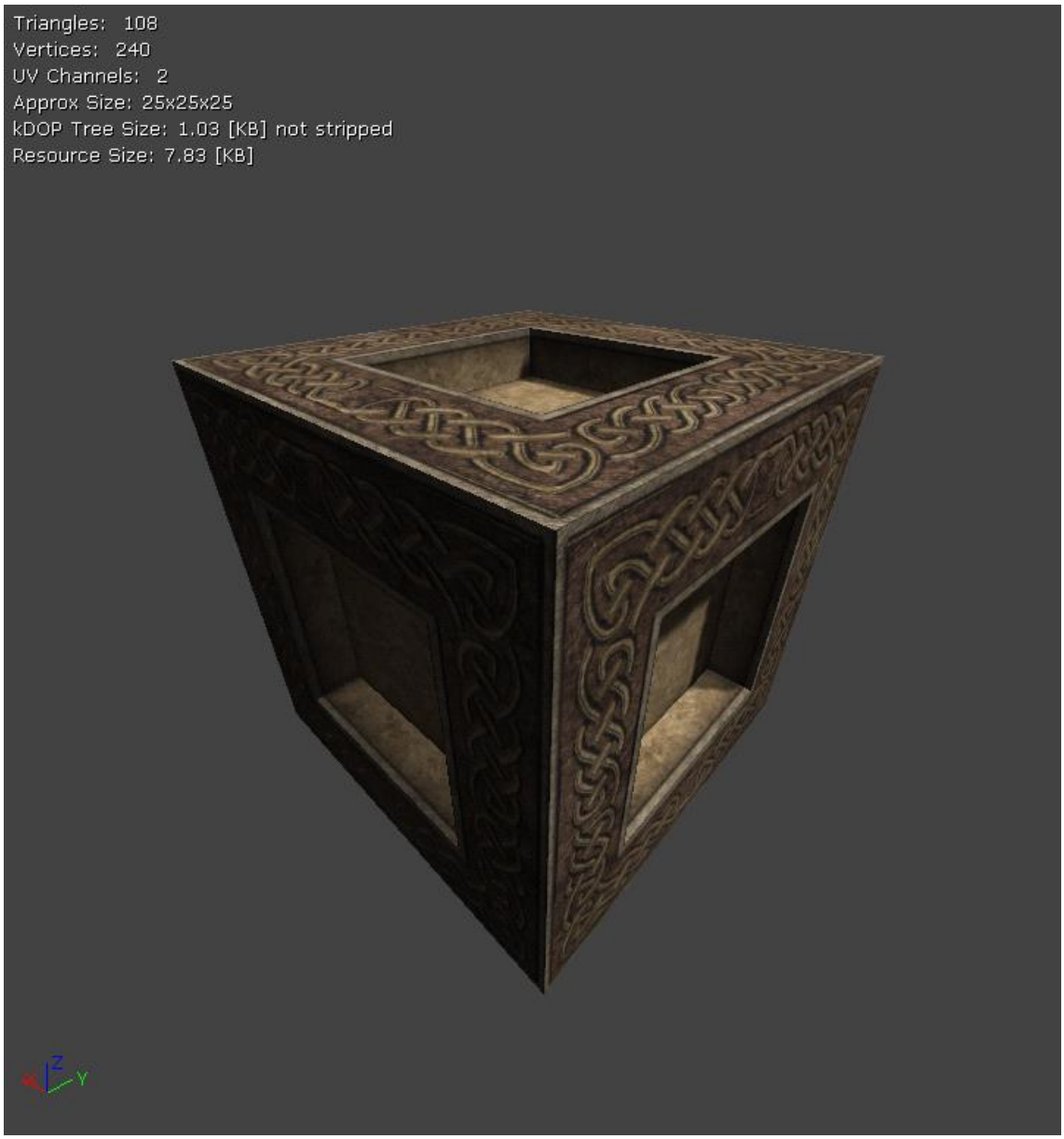

Figura 52: Static Mesh Cubo Metafísico

Static Meshs representam quase 70 % do desenvolvimento com a UDK, boa parte das peculiaridades exigidas pelas UDK para produção das mesmas já foram explicadas acima, contudo na janela de edição de *mesh* vale a pena colocar alguns pontos importantes.

Para abrir a janela de configuração de *static mesh* no UDK basta clicar duas vezes no objeto dentro do *content browser* e teremos esta janela:

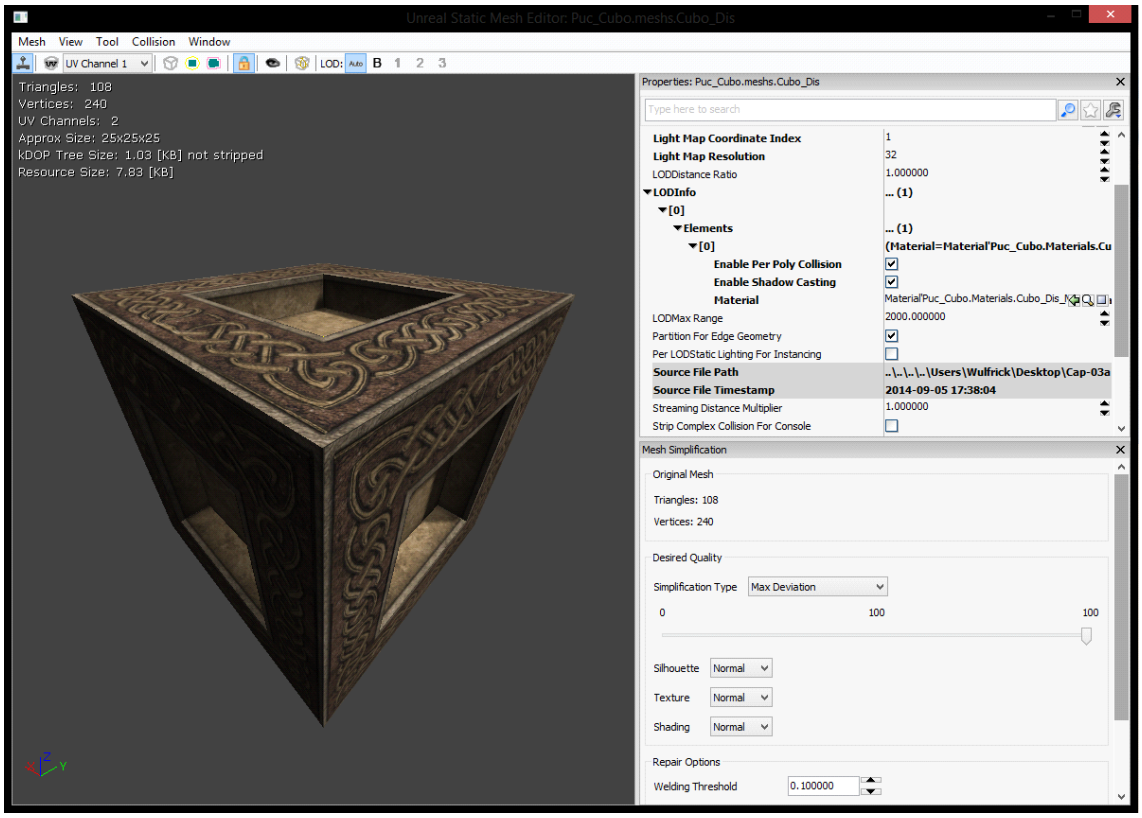

Figura 53: Janela das Configurações da Static Mesh

Nessa janela, podemos setar várias opções dentre elas as mais importantes são:

- LOD, *lightmap resolution*, seleção do UV set, escolha do material e colisão.

- LOD ou *Level of Detail* e um sistema de ação capaz de setar através da distancia o nível de detalhamento da sua *static mesh*. Esse sistema e importante porque através dele podemos economizar processamento. O LOD simplificadamente e diminuição da resolução da textura e da quantidade de polígonos do nosso objeto para formas mais simples.

- A UDK conta com um sistema periférico fornecido pela Simplaygon, capaz de gerar os LODs necessários para a chamada otimização do sistema 3D que não passa justamente de pensar e fazer o objeto possuir um comportamento que economiza processamento.

- Para gerar LODS para seu objeto pela UDK basta clicar em *mesh* e selecionar *Generate LOD*, nesse momento teremos uma janela para setar em qual faze iremos colocar nosso, LOD, a UDK trabalha com 3 fases de LOD possíveis.

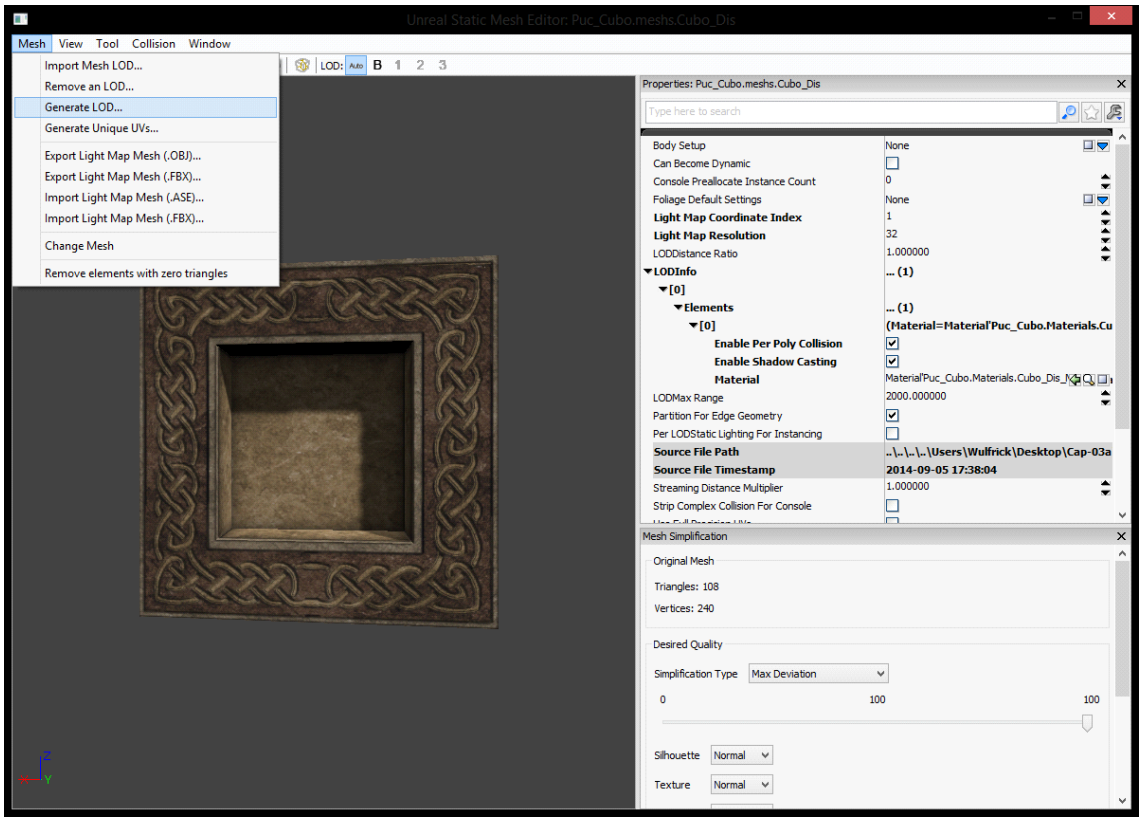

Figura 54: Setando o LOD

Após setar o local do LOD teremos uma nova janela nela poderemos setar a nova quantidade de polígonos e a distância da simplificação.

Lightmap resolution, basicamente é a resolução do mapa de luz que será gerado a partir da UV do *lightmap*. Sua qualidade depende da importância do seu modelo, no estado atual esta setado para 32 como padrão, contudo vamos alterar para 64 para obter uma melhor qualidade pensando que nosso objeto possui certo nível de importância dentro do *level design*.

Também podemos setar qual UV set esta presente nosso *lightmap* para isso basta setar o número da UV *Light Map Coordinate Index*.

Em LODinfo, teremos os materiais que compõem o objeto e nesse momento podemos setar novos materiais.

E finalmente a colisão, a colisão para os objetos, aquilo que no impede no jogo de ir para uma outra área, de continuarmos andando sem parar, podemos criar uma colisão para nosso objeto na barra superior em *Collision*, lá teremos as ferramentas para o desenvolvimento de uma colisão para o nosso objeto.

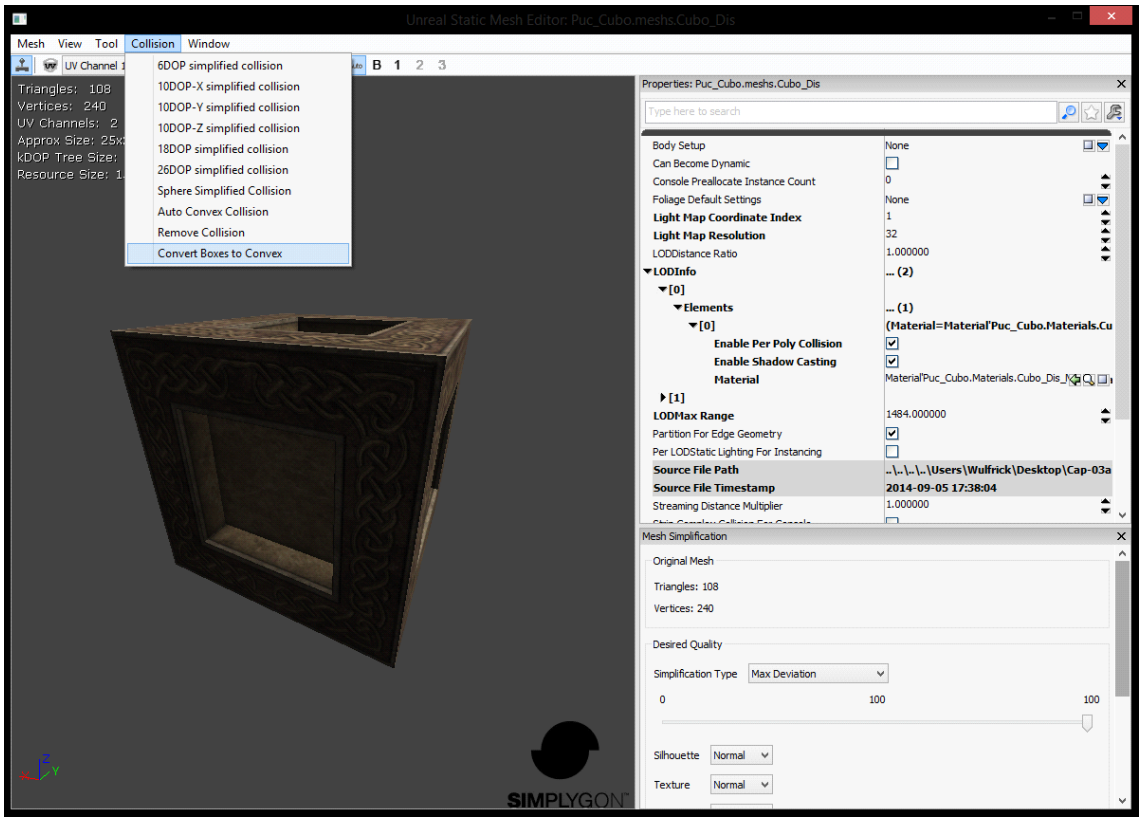

Figura 55: Collision

Contudo para obtermos uma colisão *perpoly*, ou seja, uma colisão exata com a composição poligonal do nosso objeto precisamos desativar 3 opções que basicamente setam para a colisão simplifica. Vale lembrar que uma colisão *perpoly* pede mais processamento então os dois processos têm de serem utilizados visando à importância da *mesh* com o peso no processamento.

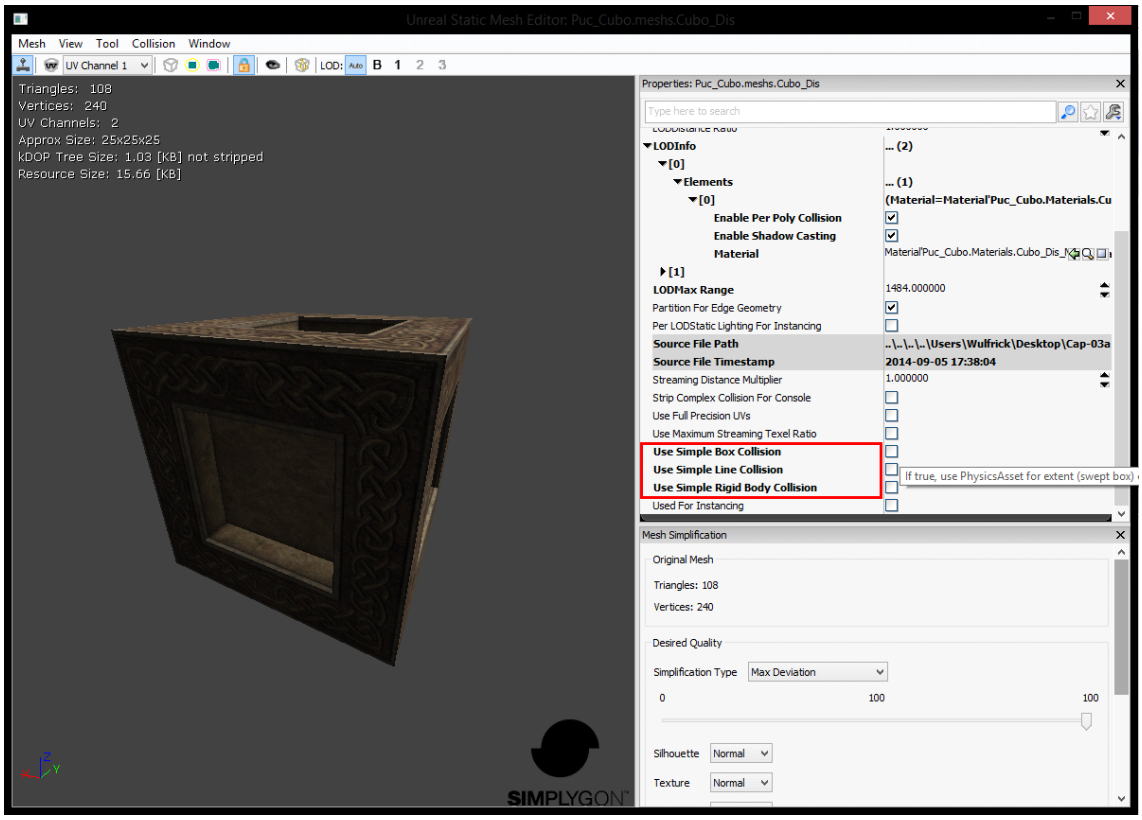

Figura 56: Collision (2)

3.5 – Material

Mesmo com o sistema de importação já criando o material do nosso objeto, às vezes é necessária a criação de um material e a edição.

Para isso vamos clicar no ícone material do *content browser* e assim abrir a edição de materiais, assim que acionamos brilho a um objeto, fazemos água no motor de jogo, dentre infinitas possibilidades. Teremos esta janela:

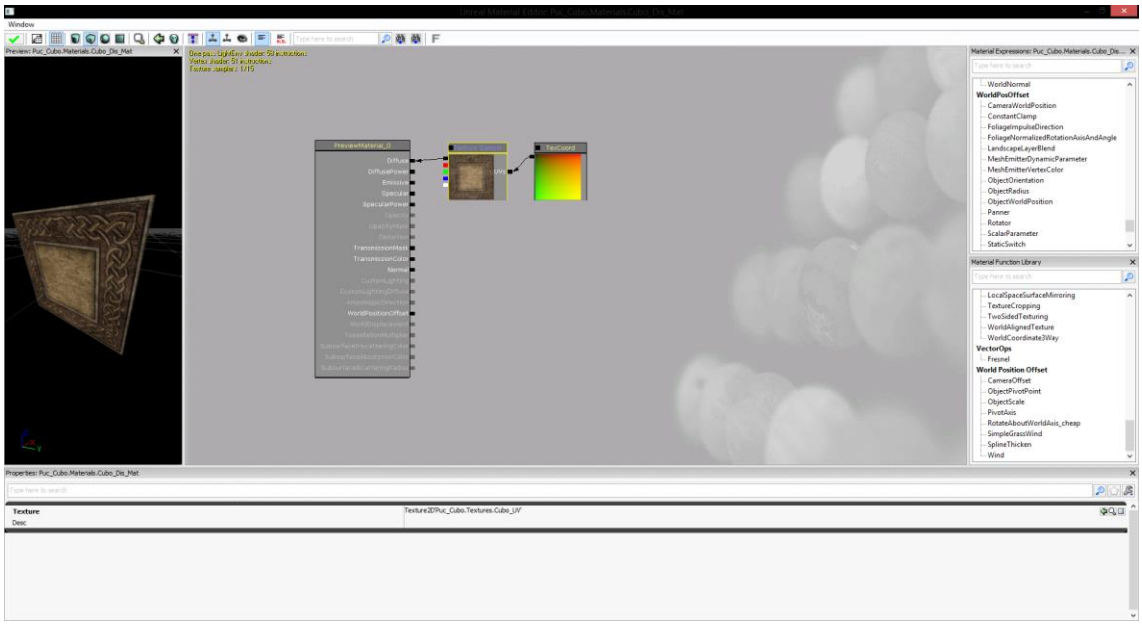

Figura 57: Material Editor

Nesta janela teremos a edição de material, por padrão encontramos essa janela já com a textura do *canal difuse* já setada.

As possibilidades de produção de material são gigantescas, mas lembrando que este é um material introdutório, e para isso vamos setar os mapas de texturas, demonstrando assim a edição de material.

Mapas de textura são imagens capazes de produzir um efeito no material do objeto, estes efeitos vão desde simular relevo no modelo 3D a aplicar um brilho específico em uma região do modelo.

Para continuar a edição do nosso material precisamos de alguns mapas de textura, para obtê-los vamos utilizar um programa externo, pode ser desde alguns conhecidos como *Crazy Bump* ou outros não tão famosos como o *Bitmap2Material*.

Para este momento vamos utilizar apenas dois mapas, um mapa de normal, possibilita simular relevo na *mesh* e um mapa *especular* para ter controle sobre seu brilho.

Após obter seus mapas seguimos o mesmo processo de importar, mas dessa vez vamos selecionar a pasta *textures*.

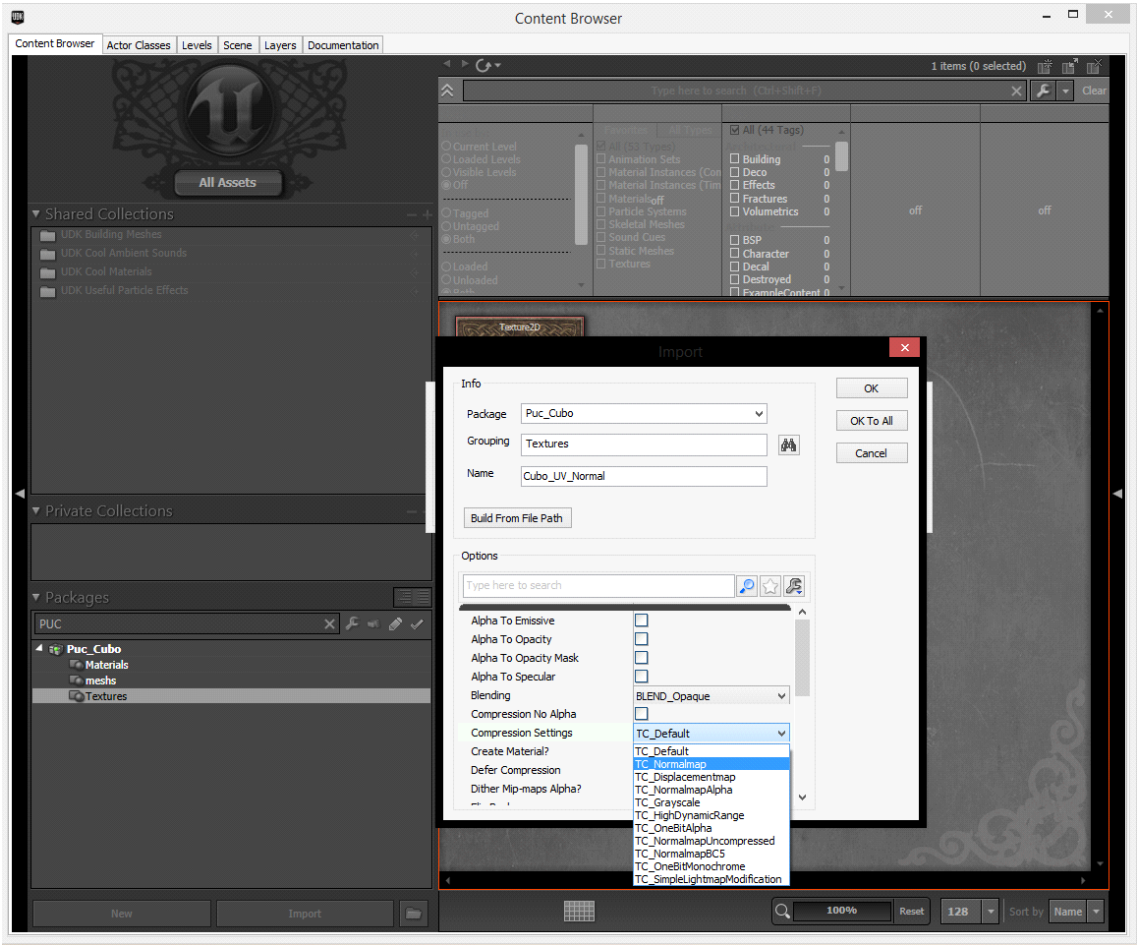

Figura 58: Importar

No momento da importação é imprescindível que se mude o tipo de compressão da textura para seu tipo específico, no caso da normal, como demonstrado acima.

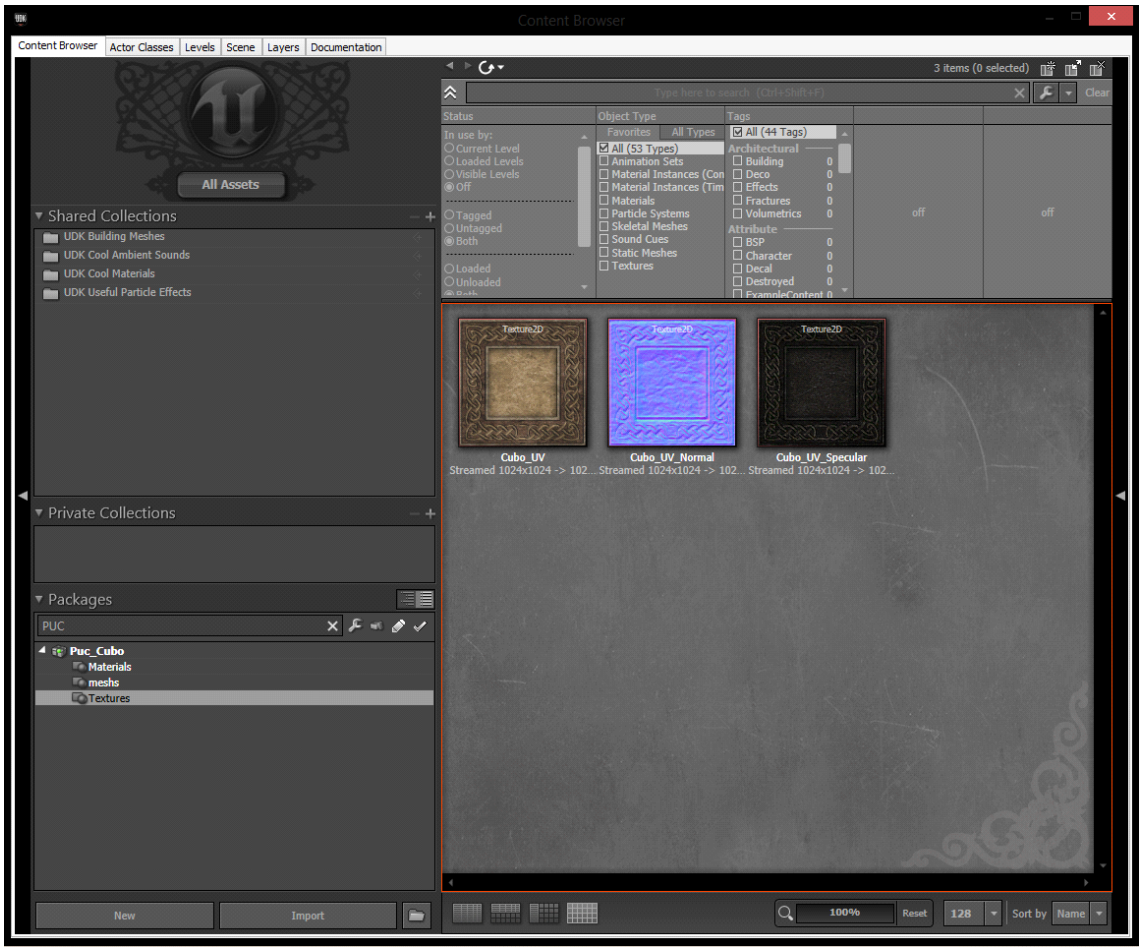

Figura 59: Mapas de Textura Importados

Pronto, agora temos dois mapas textura, vamos retomar agora a nossa edição de material.

Para trazer a textura que queremos para o nosso material precisamos primeiro selecioná-la no *content browser* apenas clicando sobre ela já com a Edição de material aberta.

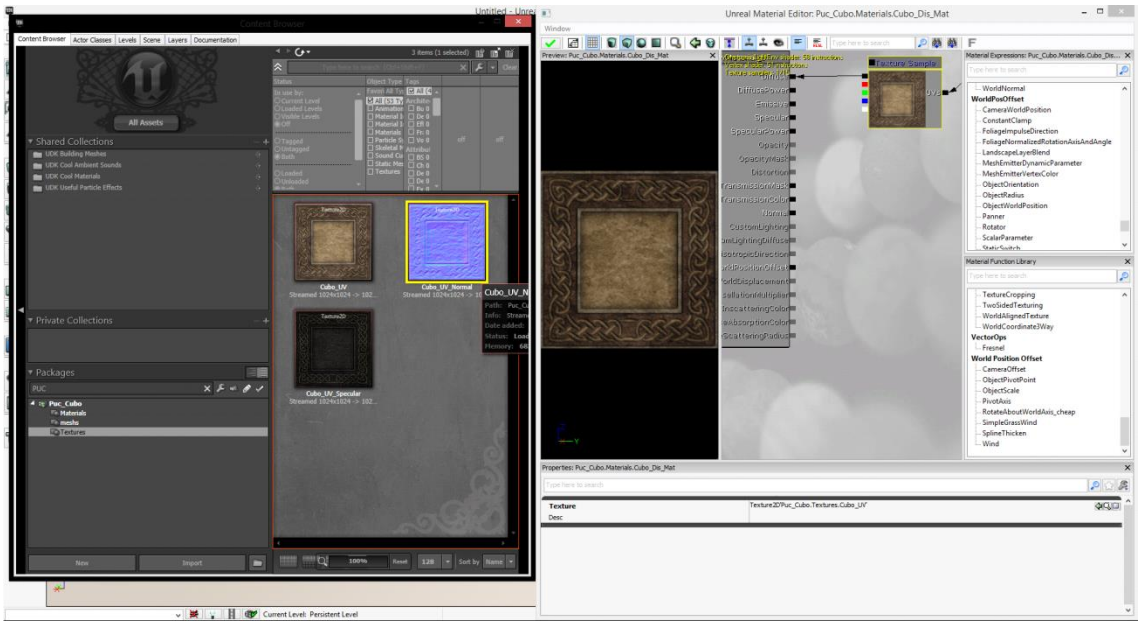

Figura 60: Trazendo a Textura para o Editor de Material

E agora segurando a Tecla *T* do teclado clicamos na área de edição do *material editor*.

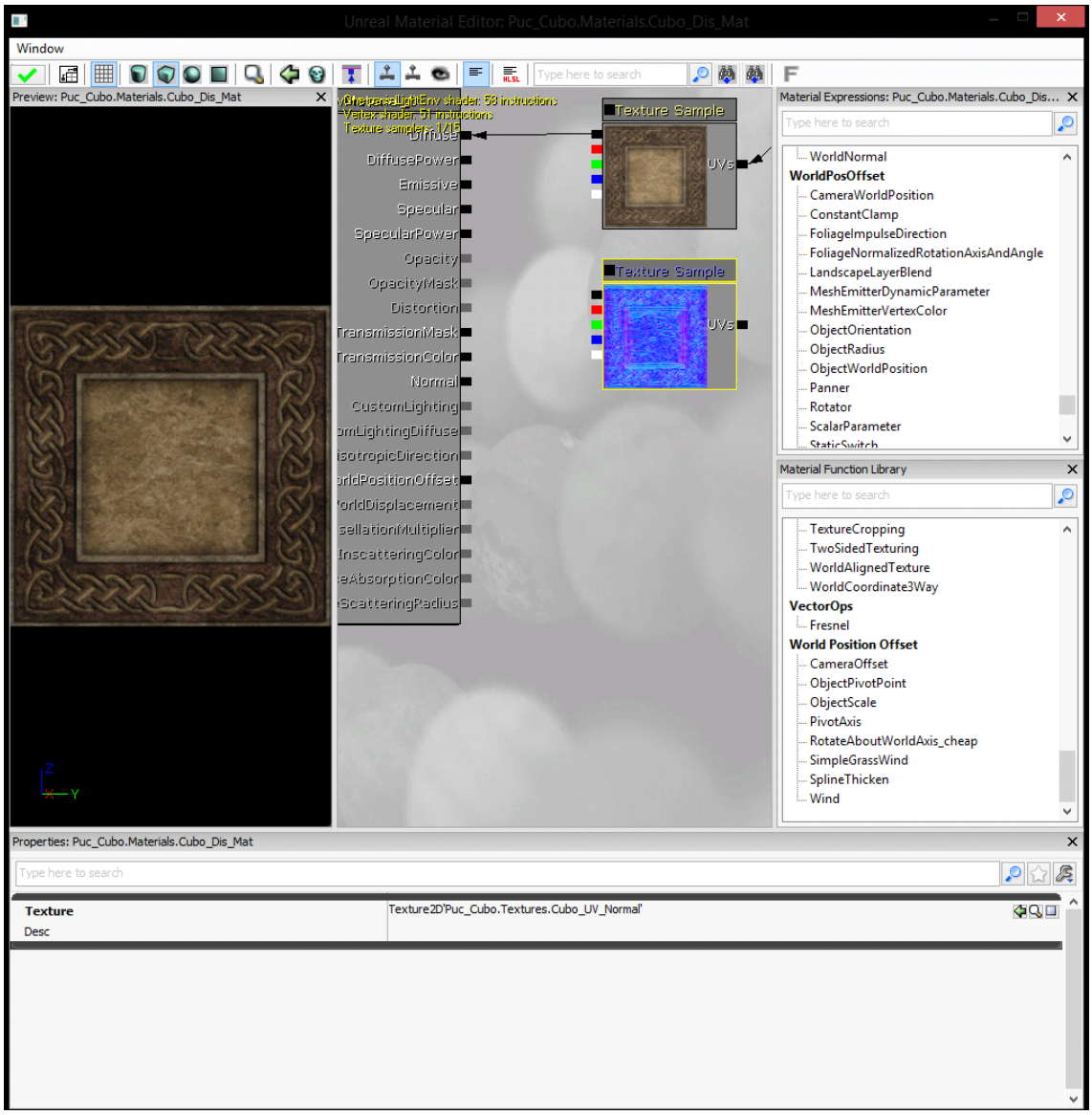

Figura 61: Textura Trazida ao Editor de Material

Pronto, nossa textura foi trazida para o editor de material, para criar nosso material resta agora apenas fazermos o *link* com seu *input* respectivo, no caso *normal maps* são conectados no *input* normal do material, aplicando assim um relevo a textura. Dando profundidade à mesma, como explicamos no capítulo 1 da presente pesquisa no item texturização, o mapa de *bump*. O mesmo processo se dá ao *specular map*, como vemos na imagem abaixo, os canais de *normal* e *specular* estão agora conectados.

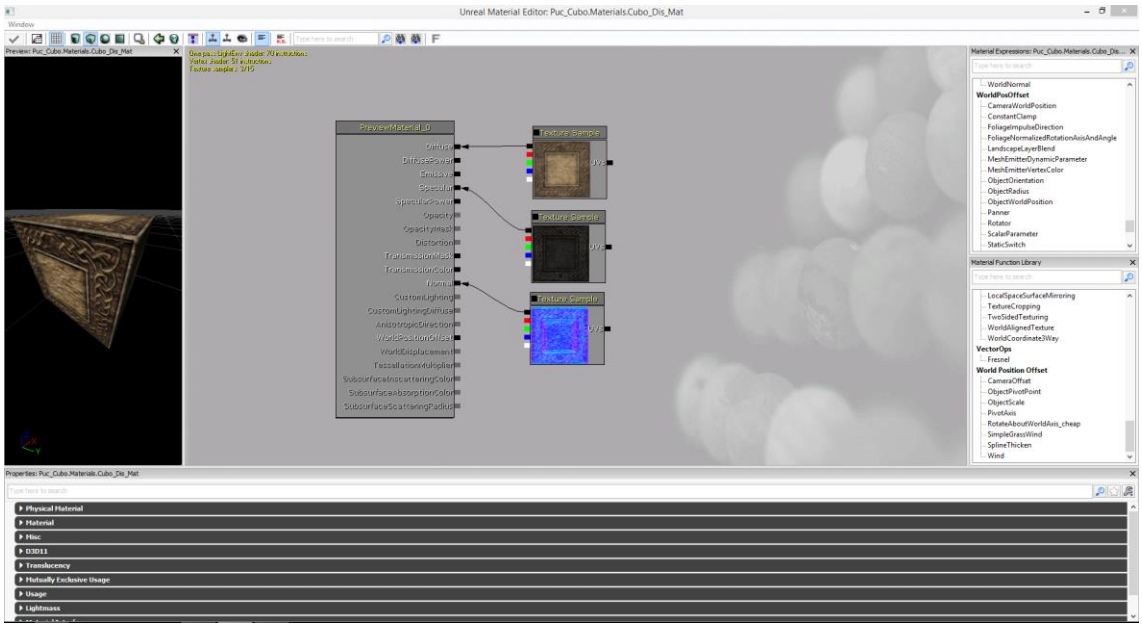

Figura 62: Mapas Conectados no Editor de Material

3.6 – Breve Level Design e Construção Modular

Ao final do processo teremos esta janela, para aplicar as modificações basta clicar no *checker* no canto superior direito no material editor. Assim terminamos a produção do nosso primeiro *asset* e podemos começar um breve *level design* com nosso objeto:

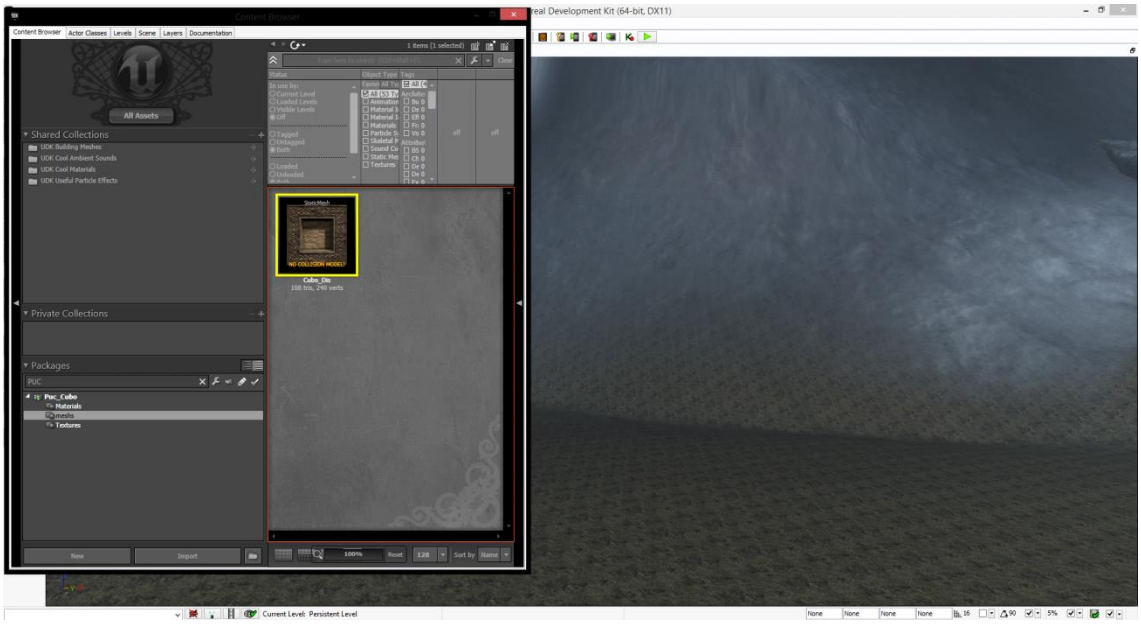

Figura 63: Selecionando Cubo

Abrindo o *content browser* indo na pasta *mesh*, vamos selecionar o nosso cubo e arrastar ele para dentro do editor.

Devido ao tamanho do Objeto pode ser que ele não apareça de imediato, nesse caso basta escalonar o objeto, aperte barra de espaço até chegar ao *gizmo* de escala, um dos cubos vermelhos irá escalonar todo o objeto simetricamente.

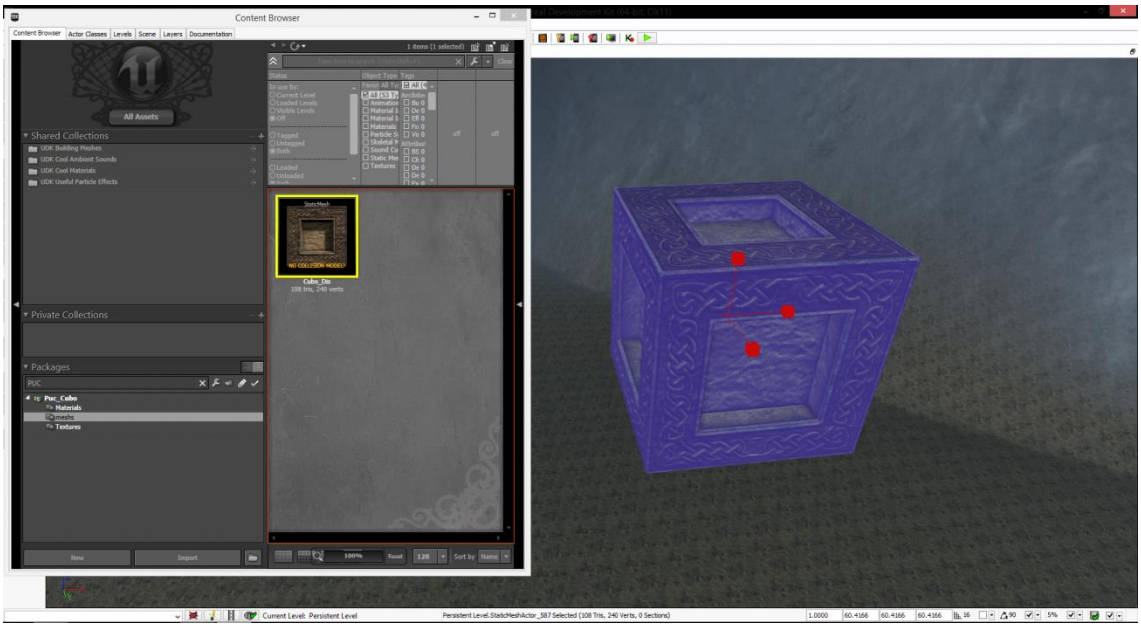

Figura 64: Cubo sendo Escalonado

Agora que já temos nosso objeto no editor podemos fechar o *content browser* e pensar uma estrutura para nosso cubo.

Nesse momento vale a pena falarmos sobre a construção modular, isso é:

Em vez de modelarmos uma casa completa e trazer ela para o UDK, podemos modelar telhados portas e paredes e com vários tipos de texturas e modelos chegar a muitas combinações diferentes possibilitando um melhor aproveitamento de processamento, estética e tempo.

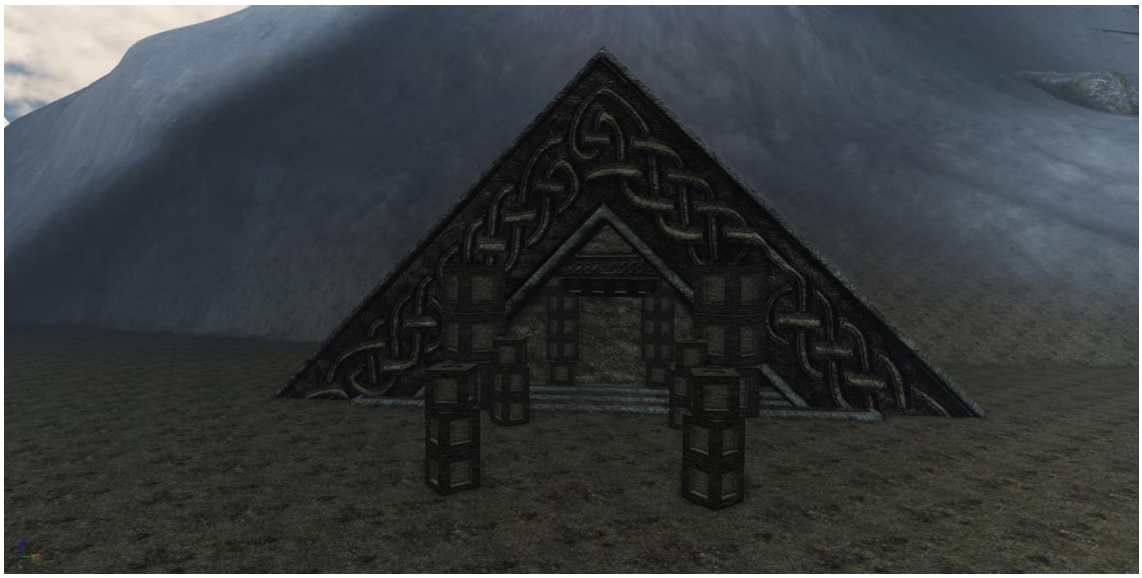

Figura 65: Construção Modular a partir do Cubo

Utilizando apenas o cubo em um sistema modular, podemos ter muitas possibilidades de construção esta foi uma delas.

3.7 – Demais Aplicações

O presente tutorial pode ser utilizado para diferentes fins de produção em UDK, vamos explicitar um *case* estritamente relacionado com nossa pesquisa, tal qual foi desenvolvido juntamente com a presente pesquisa.

Desenvolvemos um artigo intitulado: *Desenhar, Modelar e Design de Nível Como Processos Ontológicos na Produção de Jogos Digitais* (Trentin; Marques; Petry; Hildebrand; Gatti, 2014), que foi apresentado em um evento de *games*, o Gamepad que ocorreu na Feevale, em Novo Hamburgo, Rio Grande do Sul em maio de 2014, o artigo tem como ideia geral apresentar uma reflexão sobre os processos técnicos artísticos da produção de jogos como *concept art*, modelagem 3D e *level design*, para tal utilizamos como *case* modelar, um projeto que estamos desenvolvendo em conjunto, intitulado *A Ilha dos Mortos*, baseado na pintura do pintor simbolista Arnold Böcklin, o projeto foi inicialmente estava se desenvolvendo em UDK, mas passará para *Unreal 4*.

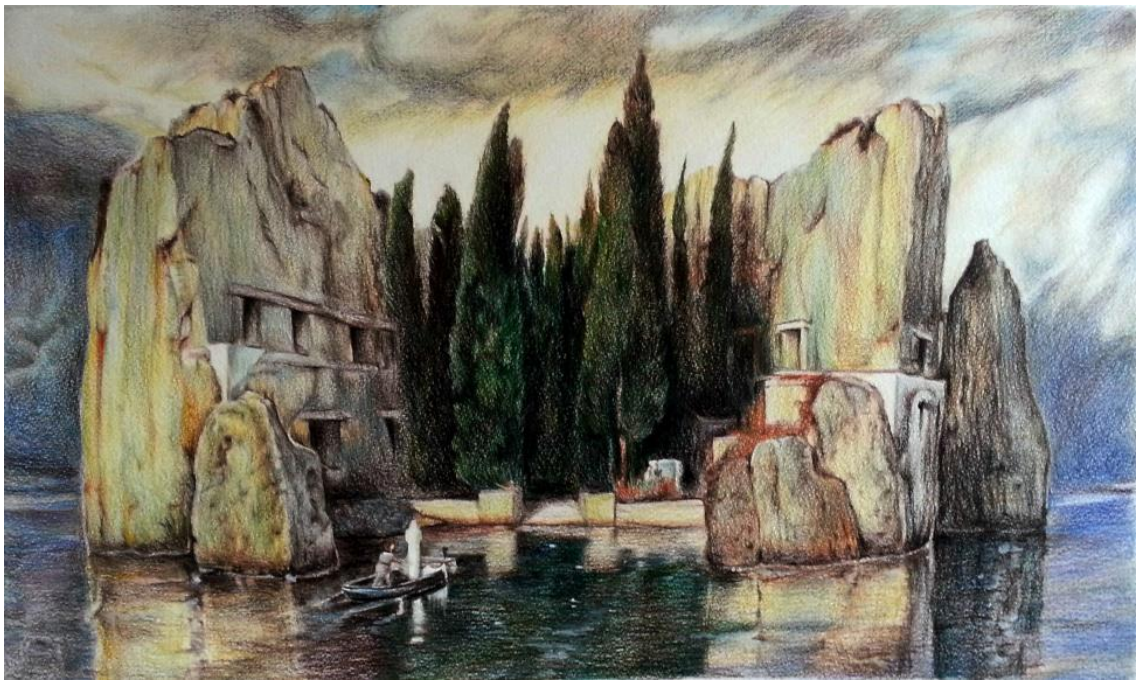

Figura 66: Pintura a lápis de cor "A Ilha dos Mortos" por Silvia Trentin

Com base no *concept art* acima feito por Silvia Trentin, desenvolvemos a versão em UDK, como podemos ver na imagem abaixo, porém não é uma versão definitiva, sim um conceito inicial, o que se chama de *X-Statement*, uma etapa de produção largamente utilizada na indústria de jogos:

> "Em uma primeira abordagem inicial do trabalho pesquisamos os conceitos técnicos desenvolvidos pela empresa Eletronic Arts, chamado de X-Statement, principalmente na abordagem que fez dele a equipe do jogo Dead Space, o qual consiste na construção básica de um momento privilegiado do jogo, representando o comportamento do jogo, sua mecânica, parte de sua atmosfera e, principalmente, a sua identidade visual. Originalmente a ideia de X-Statement foi concebida como um conceito operacional da indústria, visando a organização do projeto para o pitch (apresentação) da reunião da luz verde com os investidores, e consistia em uma apresentação ou demonstração concisa do jogo com os elementos fundamentais, capaz de despertar o interesse dos membros da reunião 20. Para Simeon Pashley, o X-Statement é definido pela produção de uma apresentação concisa que resuma o jogo de forma clara, na qual se mostre a essência do jogo a ser produzido, tendo em vista expectativas realista. O X-Statement funcionaria também como uma linha mestra que depois deveria ser seguida no desenvolvimento. Mesmo assim, o conceito e estratégia visava aqui ainda o pitch da reunião da luz verde." (Trentin; Marques; Petry; Hildebrand; Gatti, 2014, p. p. 17)

Figura 67: X-Statement da Ilha dos Mortos

Como podemos observar, a vegetação nativa da ilha é composta por ciprestes, para desenvolver os mesmos utilizamos de uma estratégia de modelagem diferenciada, inspirada no desenvolvimento e construção biológica da Divina Proporção (Proporção Áurea), dada pelas séries de *Fibonacci*, encontrada em diversas formas vivas como, por exemplo, o crustáceo Náutilus.

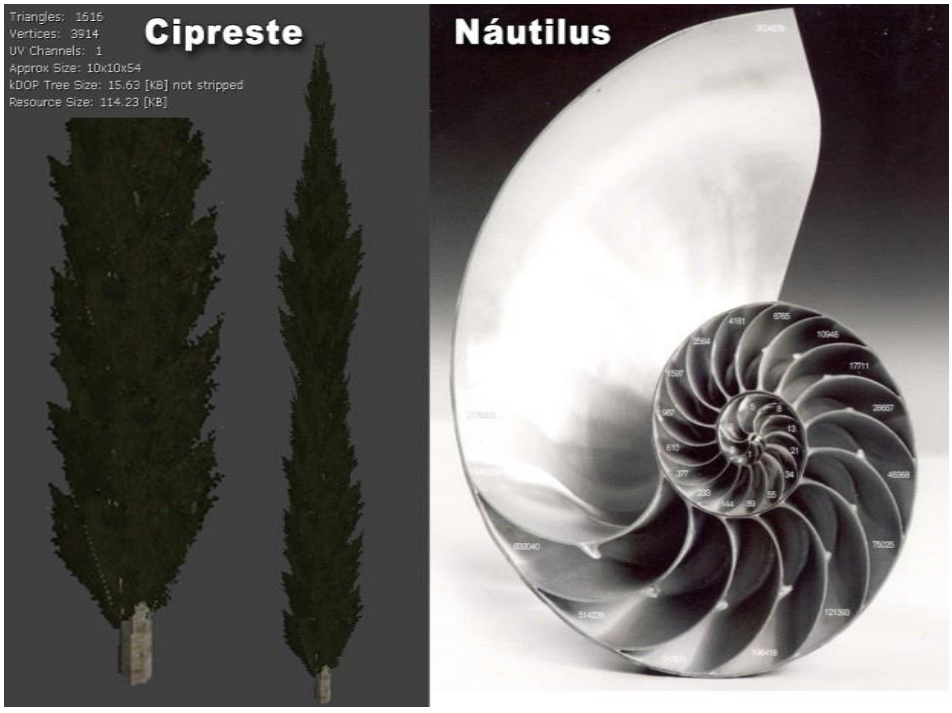

Figura 68: Cipreste e Náutilus (Trentin; Marques; Petry; Hildebrand; Gatti, 2014, p. 21)

> "A distribuição das câmaras do Náutilus serviu para a distribuição dos ramos que compõe as folhagens dos Ciprestes. Das várias construções pesquisadas e abordadas na modelagem, esse tipo de distribuição orgânica foi a que mais se aproximou, tanto da distribuição natural, quanto da distribuição representada por Böcklin." (Trentin; Marques; Petry; Hildebrand; Gatii, 2014, p. 21).

3.8 - Landscape, Kismet e Speedtree

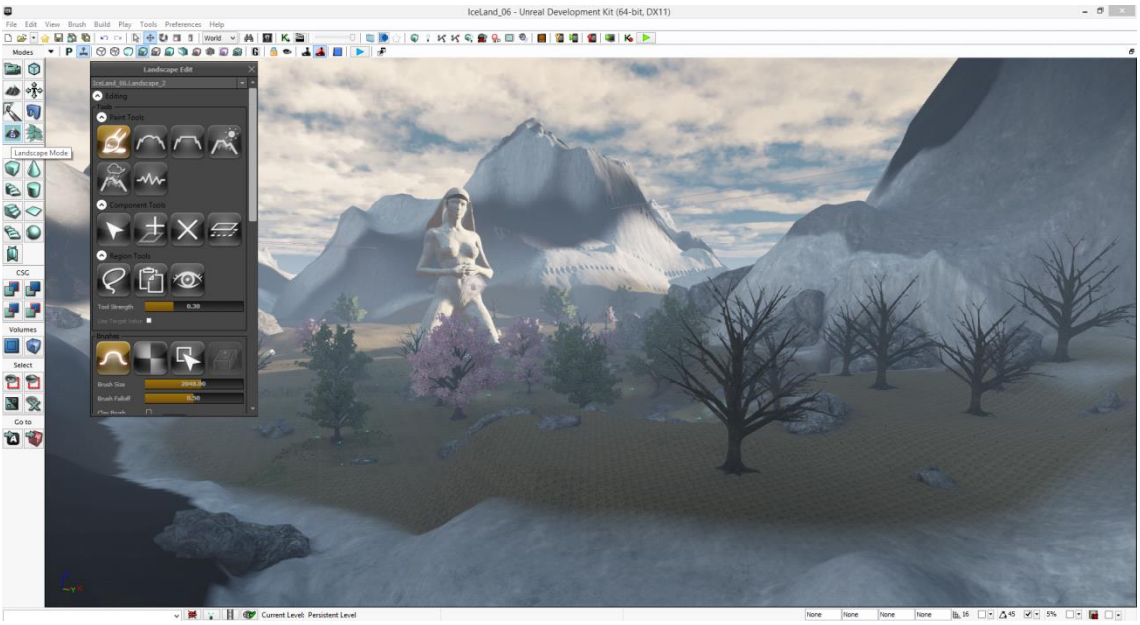

Figura 69: Landscape

No princípio da pesquisa, nós pensamos em introduzir também três ferramentas para produção em UDK, o *Landscape*, o *Kismet* e o *Speedtree*. *Landscape* é uma forma de projeção topológica em uma malha 3D dentro da UDK, contudo, nos avanços da pesquisa nos demos conta que por sua complexidade não poderia ser introduzido dentro do trabalho, contudo as etapas anteriores possuem conceitos fundamentais para a produção de um *Landscape*, sendo assim nos focamos mais nas pesquisas da produção e da *pipeline* do modelador para a UDK.

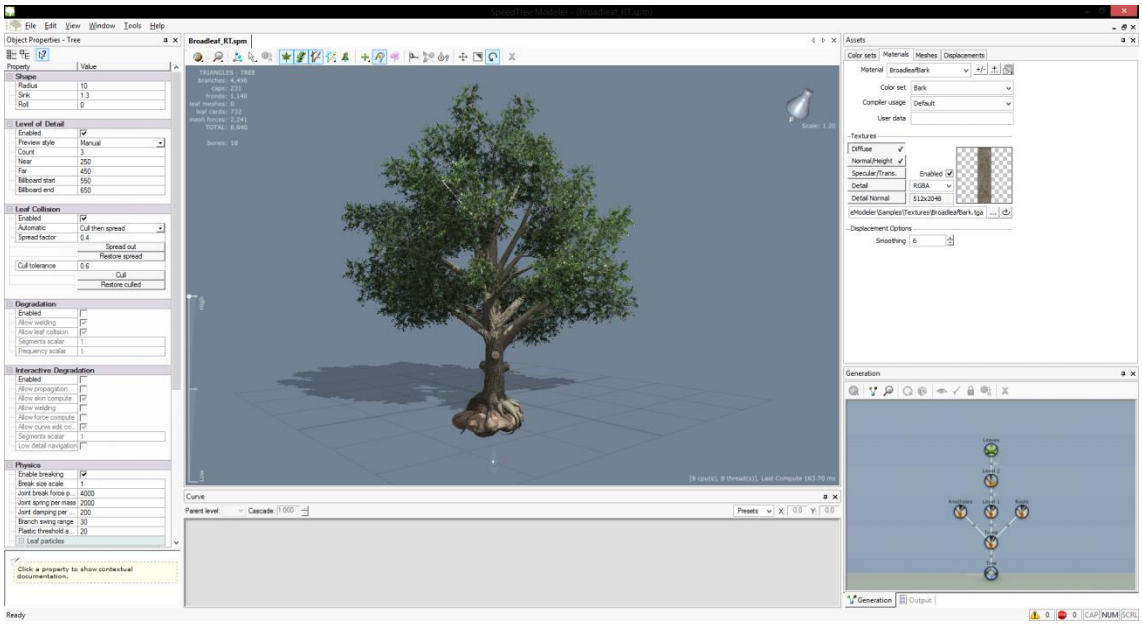

Figura 70: Speedtree

Não diferente do *landscape* o *speedtree*, editor externo para produção de árvores dentro do UDK por um processo mais elaborado e com funções de simulação avançadas, também, fora pensado em ser introduzido nesta pesquisa, contudo voltamos ao mesmo panorama do *landscape*.

3.9 – Prints do Nosso Ambiente Navegável Construído para a Pesquisa

Nessa parte selecionamos 12 (doze) *prints* do nosso ambiente navegável pronto, mesmo foi desenvolvido ao longo da presente pesquisa, procuramos manter como aspecto primordial a experiência estética (Bateman, 2012).

Figura 71: Print 1

Figura 73: Print 2

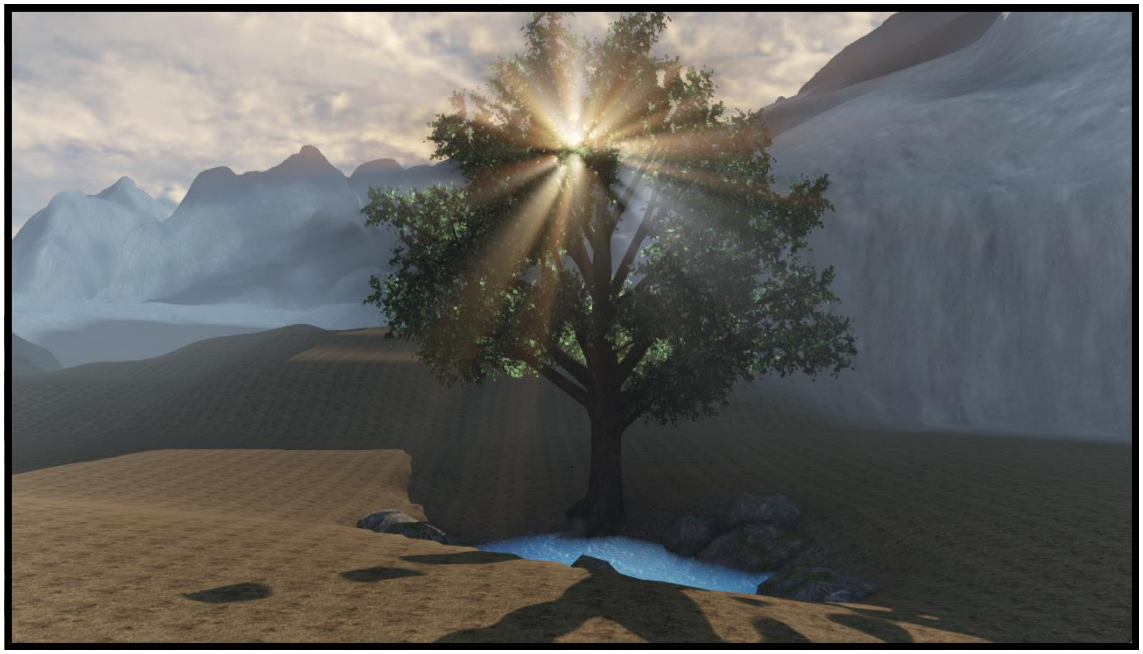

Figura 72: Print 3

Figura 75: Print 4

Figura 74: Print 5

Figura 76: Print 6

Figura 77: Print 7

Figura 78: Print 8

Figura 79: Print 9

Figura 80: Print 10

Figura 81: Print 11

Figura 82: Print 12

Conclusão

Figura 83: Cristais, Luz e Refração

Para este trabalho, organizar um material de forma introdutória, de modo que se comporte como um tutorial, mas que possua um aporte mais simplificado para aqueles que estão começando foi particularmente difícil.

Transformamo-nos à medida que crescemos intelectualmente, e depois de algum tempo de tanto pensar e executar uma tarefa simplificamos o fazer e o pensar, tal qual as suas dificuldades e elementos cruciais acabam caindo na trivialidade que antes foram uma dificuldade.

Produzir um material para alguém que possivelmente nunca desenvolveu nenhum *game* ou ambiente navegável, e tentar percorrer um antigo caminho como se tudo fosse novo; A principal justificativa da minha pesquisa sempre foi visando sanar um problema comum na comunidade de jogos quando livros quando ensinam tomam conhecimentos específicos como preceito esquecendo que tal conhecimento é

necessário para o avanço do usuário daquele material, ou acabamos ensinando o usuário apenas a utilizar caminhos sem que este entenda seus porquês. Eu encerro esse trabalho contando com sua evolução.

O trajeto da pesquisa percorrendo particularmente os conhecimentos específicos que o motor de jogo requer e a escassez dessas informações para iniciantes foi longo e trabalhoso.

É gratificante ver o trabalho já com alguma aplicação ajudando pessoas de mais próximas no desenvolvimento de seus projetos.

Tive uma surpresa agradável de observar um crescimento da utilização da UDK mesmo com o lançamento de seu sucessor mais robusto. A comunidade ainda precisara deste conhecimento por mais algum tempo e acredito que ele ainda será valido como uma introdução também a *Unreal 4*.

Figura 84: Unreal 4

Fico um pouco triste em encerrar este trabalho sem poder trabalhar com *landscape* e *speed Tree*, contudo me dei conta que não poderia introduzir eles justamente porque eles necessitam de conhecimentos avançados e este trabalho seguindo sua proposta atual não poderia comporta-los. Talvez em um breve futuro com a continuidade das pesquisas visando o desenvolvimento de projetos como *Phoenix Wind* de Cristiano Toneis, *Ilha Cabu* de Arlete e Luís Carlos Petry e a *Ilha dos Mortos* pela PUC-SP, eu possa construir uma evolução agora passando para elementos mais elaborados do desenvolvimento

utilizando a Tecnologia *Unreal*.

Figura 85: Logo da Ilha Cabu

Figura 86: Pintura A Ilha dos Mortos de Arnold Böcklin e X-Statement da Ilha dos Mortos por Gabriel C. Marques

Podemos concluir com base nas exemplificações dadas (dentre algumas podemos citar o exemplo do *X-Statement* da *Ilha dos Mortos*, por exemplo, mesmo sendo apenas um trabalho inicial) na presente dissertação e com base em nosso tutorial, que podemos sim desenvolver um material de qualidade gráfica e estética superior, com padrão de indústria utilizando o motor de jogos UDK. Mesmo que a equipe de produção não seja grande é possível, mostramos os *prints* de nosso ambiente navegável feito pelo presente

pesquisador, também exemplificamos em nosso Capítulo 2 *UDK e sua História* alguns jogos de qualidade *Triple A* produzidos com o motor de jogo, dentre eles citamos *Alice Madness Returns*, que apesar de ser um jogo *Triple A*, foi produzido por 13 pessoas, que é um número consideravelmente pequeno, mesmo assim ganhou o prêmio de melhor direção de arte (como já explicitado com maiores detalhes no capítulo 2 da presente pesquisa de mestrado). Então sim é possível.

Isso forma um terreno muito fértil e produtivo de pesquisas com UDK e tecnologia *Unreal* em geral.

Por conta disso, temos como planejamento que a pesquisa se estenda de certa forma para o doutorado, dizemos de certa forma porque pretendemos dar um passo adiante e desenvolver uma pesquisa sobre desenvolvimento em *Unreal 4*, por ser um motor de jogo extremamente novo (lançado em março de 2014), ainda não há livros disponíveis sobre o assunto, também há muito pouco material na internet, (em comparação com o que se tem de versões anteriores da *Unreal*) por ora é um terreno praticamente inexplorado, o que é muito natural, dado que a tecnologia *Unreal 4* tem apenas poucos meses.

Assim pretendemos pesquisar *Unreal 4* no doutorado, transpor a *Ilha Cabu* para *Unreal 4* e continuar o projeto da *Ilha dos Mortos*.

É importante que contribuamos para mudar um panorama bastante comum no Brasil que é o de se desenvolver apenas jogos para celulares e/ou *tablets* , esse tipo de pesquisa contribui para desmistificação que se desenvolveu na cultura de nível nacional de que é impossível produzir algo de qualidade *Triple A*, para console ou PC em um país como o Brasil, isso em absoluto não é verdade, com o advento dos jogos digitais cada vez mais realistas, esse panorama nacional tem de ser mudado.

Uma boa maneira de realizar uma contribuição nesse aspecto é investir e acreditar em pesquisas de desenvolvimento de jogos digitais, outra maneira é produzi-las, de forma que nosso trabalho não acaba aqui no final dessa conclusão, melhor dizendo ele apenas está começando.

Glossário

1 - ***Assets***: São os materiais que compõem o jogo, podendo ser assets de arte como modelagem 3D e textura ou assets de programação como códigos.

2 - ***Arte de conceito***: Tradução para concept art. Nada mais é do que as ilustrações prévias (finalizadas ou não) à modelagem 3D. É o guia para o modelador 3D.

3 - ***Box Modeling***: Trata-se de uma técnica de modelagem que consiste em adicionar linhas (edges) formando assim mais faces para serem puxadas.

4 - ***Brainstorm***: Traduzindo seria tempestade cerebral. Trata-se de um momento de intensa criatividade onde se discute o roteiro do jogo, guia de estilo do mesmo, tal como outras ideias a serem integradas ou não no projeto.

5 - ***Cutscene***: Trata-se de uma sequência que o jogador não tem controle, porém ela ocorre sem interrupção, ou seja, não é uma CG. É sempre feita com elementos do próprio jogo.

6 - ***Design de Nível***: Tradução para o termo level design. Trata-se da "montagem" das fases ou níveis do jogo.

7 - ***Game Designer***: Profissional que trabalha com design de jogos.

8 - ***Gameplay***: É a parte ativa do jogo, ou seja, tudo aquilo que é jogável, onde há interação.

9 - ***Gizmo***: Orientação no eixo X, Y, Z, para movimentação, escalonamento e rotação.

10 - ***Modelagem 3D***: Existem jogos 2D e 3D. Os jogos 3D contém modelos tridimensionais, ou seja, que podem ser vistos pelas 3 dimensões, a modelagem 3D é a confecção desses modelos.

11 - ***Box Modeling***: Trata-se de uma técnica de modelagem que consiste em adicionar linhas (edges) formando assim mais faces para serem puxadas.

12 - ***Modelagem Poly By Poly***: É uma forma de se modelar em 3D que consiste em modelar puxando uma face de outra face do objeto.

13 - ***Motor de Jogo***: É o software no qual o design de nível do jogo é construído, é no motor de jogo que se "monta o jogo", após prontas as modelagens 3D.

14 - ***Publisher***: É a produtora do jogo, a que lança o jogo.

15 - ***Sourcecode***: Traduzindo para código fonte. O sourcecode é a fundação de um software, sua base.

16 - ***Storyboard***: Funciona como um organizador gráfico composto por uma série de ilustrações do jogo. Também é utilizado no cinema.

17 - ***Textura***: É o que envolve os modelos 3D, o que da cor aos mesmos. Sem textura o jogo não teria cor.

18 - ***Triple A***: É uma nominação que designa jogos de grande qualidade gráfica, de alto padrão de qualidade.

Bibliografia

BATEMAN, C. (2012). *Implicit Game Aesthetics (1): Crawford's Taxonomy*. In: InternacionalIhobo. Disponível em http://blog.ihobo.com/2012/04/implicit-game-aesthetics-1-crawfords-taxonomy.html. Acesso em fevereiro de 2014;

BOBANY, A. (2007). *Video Game Art*. Teresópolis. Novas Ideias;

BUSBY, J; PARRISH, Z; WILSON,J. (2009). *Mastering Unreal Technology Volume 1: Introduction to Level Design With Unreal Engine 3*. Indianapolis. Editora Sams;

_____. (2009). *Mastering Unreal Technology Volume 2: Advanced Level Design Concepts With Unreal Engine 3*. Indianapolis. Editora Sams;

CARDOSO, Rafael. (2008). *Uma Introdução a História do Design*. São Paulo. Editora Edgar Blucher;

CHANDLER, H. M. (2012). *Manual de Produção de Jogos Digitais*. Porto Alegre. Editora Bookman;

FINCH, A. (2013). *The Unreal Game Engine: A Comprehensive Guide to Creating Playable Levels*. United Kingdom. 3DTotalPublishing;

HEIM, M. H. (1994). *The Metaphysics of Virtual Reality*. Nova York. Oxford USA Trade;

HUIZINGA, J. (2001). *Homo Ludens: O Jogo Como Elemento da Cultura*. São Paulo. Editora da Universidade de São Paulo;

LAUREL, B. (1993).*Computer as Theater*. EUA. Addison-Wesley Professional;

MANOVICH, L. (2001). *The Language of New Media*. Cambridge: The MIT Press;

MARQUES, G. C; PETRY, L.C. (2014). *A Influência da Pintura Barroca na Produção do Light Design nos Jogos Digitais: Estudo de Caso Dead Space 2*. Porto Alegre. SBGames 2014. Track de Arte e Design. Disponível em: http://www.sbgames.org/sbgames2014/files/papers/art_design/full/A&D_Full_A%20Influencia%20da%20Pintura%20Barroca.pdf . Acesso em 12/2014. ISSN: 2179-2259;

MCGEE, A. (2011). *The Art of Alice Madness Returns.* Milwalkie. Dark Horse;

MOONEY, T. (2012). *Unreal Development Kit Game Design Cookbook*. Livery Place. Packt Publishing;

MOORE, R. (2011). *Unreal Development Kit Beginner's Guide*. Livery Place. Packt Publishing;

MORIN, E. (1990). Introdução ao Pensamento Complexo. Porto Alegre. Editora Sulina;

MURRAY, J. (2003). *Hamlet no Hollodeck: O Futuro da Narrativa no Ciberespaço*. São Paulo. Editora Unesp;

NOVAK, J. (2010). *Desenvolvimento de Games*. São Paulo. Editora Cengage Learning;

_____. (2011). *Game Development Essentials: An Introduction*. New York. Delmar Cengage Learning;

PARDEW, L. (2004). *Begining Illustration and Storyboarding for Games*. EUA.

Cengage Learning PTR;

_____. (2005). *Basic Drawing for Games*. EUA Cengage Learning PTR;

PETRY, L. C. (2003). *Topofilosofia: O Pensamento Tridimensional na Hipermídia*. Tese de doutorado defendida no Programa de Pós Graduação em Comunicação e Semiótica da Pontifícia Universidade Católica de São Paulo. Orientador: Sérgio Bairon;

_____; LOPES, E; PONTUSCHKA, M.; FRAGOSO, F.; MARQUES, G.; ZANSAVIO, W.H.I. (2012). *Organizando os Mapas de Iluminação dos Assets de Arte para os Motores de Jogos: Considerações Metodológicas Para o Caso da Produção*

Voltada ao Motor de Jogos UDK in BRANCO, M. *et al* (Org.). (2012). Jogos Eletrônicos na Prática: Livro de Tutoriais. Novo Hamburgo. Feevale;

_____; COSTA, T. S.; SILVA, A. V.; MARQUES, G. C.; CASARINI, M. (2013). *Wasteland Beautiful: o estatuto ontológico da imagem nos mundos tridimensionais dos games.* Revista Obra Digital da Universidade de VIC. Barcelona. Disponível em: http://revistesdigitals.uvic.cat/index.php/obradigital/article/view/26. ISSN: 2014503;

RABIN, S. (2012). *Introdução ao Desenvolvimento de Games Vol I*. São Paulo. Cengage Learning;

_____. (2013). *Introdução ao Desenvolvimento de Games Vol III.* São Paulo. Cengage Learning;

SCHELL, J. (2010). *A Arte do Game Design: O Livro Original*. Rio de Janeiro. Editora Elsevier;

SCHUYTEMA, P. (2008). *Design de Games: Uma Abordagem Prática*. São Paulo. Cengage Learning;

SENNETT, R. (2009). *O Artífice.* São Paulo. Editora Record;

STONEHAM, B. (2010). *How to Create Fantasy Art for Video Games*. Londres. Barron's;

THORN, A. (2011). *UDK Game Development*. Boston. Course Tchnology Cengage Learning;

TRENTIN, S.; MARQUES, G.C.; PETRY, L.C.; HILDEBRAND, H. R.; GATTI, D. C.; (2014). *Desenhar, Modelar e Design de Nível Como Processos Ontológicos na Produção de Jogos Digitais.* Novo Hamburgo. Gamepad Universidade Feevale. (No Prelo);

YIN, W. (2011). *Impeccable Scene Design*. Berkeley. Gingko Press;

ZÖLLNER, F. (2005). *Leonardo Da Vinci: Sketches & Drawings*. Florença. Editora Taschen;

Ludografia

MCGEE, A. (2011). *Alice Madness Returns*. Eletronic Arts. Versão para PC. Disponível em: http://www.ea.com/alice. Acessado em: 07/2014;

WARNER BROS. (2013). *Batman Arkham Origins*. Warner Bros. Versão para PC. Disponível em: https://www.batmanarkhamorigins.com/pt-br . Acessado em 11/2014;

LEVINE, K. (2013). *Bioshock Infinite*. 2K Games. Versão para PC. Disponível em: https://www.bioshockinfinite.com/?RET=&ag=true . Acessado em 11/2014;