immerso

Experiências em realidade virtual baseadas
em grandes obras da literatura

IMMERSO

Experiências em realidade virtual baseadas em grandes obras da literatura

Trabalho de Conclusão de Curso
Comunicação Visual Design - UFRJ
Bernardo Tadeu Miranda de Moura
DRE 113043725

Orientação: André Ramos
Rio de Janeiro, 07 de maio de 2018.

RESUMO

“IMMERSO: Experiências em realidade virtual baseadas em grandes obras da literatura” é um projeto que se propõe a demonstrar o processo de produção de um ambiente fotorrealista em realidade virtual, criando também um exemplo prático: uma ambientação baseada no conto “Os assassinatos da Rue Morgue”, de Edgar Allan Poe e a propor uma maneira de visualizar e compartilhar esse resultado: o aplicativo “immerso”. Seu principal objetivo, além de explicar o que deve se levar em consideração ao produzir tal ambiente e de propor maneiras de resolver os problemas que surgem durante o processo, é também analisar como essa nova tecnologia se apresenta como uma nova mídia e uma nova forma de entretenimento.

ABSTRACT

“IMMERSO: Experiences in virtual reality based on famous works of literature” is a project that proposes itself to demonstrate the production process of a photorealistic enviroment in virtual reality, and also creating a practical example: an environment based on the tale “Murders in the Rue Morgue”, by Edgar Allan Poe and to propose a way of visualizing and sharing the product: the app “immerso”. Its primary objective, besides explaining what must be taken into account when creating such environment and proposing ways of solving problems that arise during the process, is to also analyze how this new technology presents itself as a new midia and a new form of entertainment.

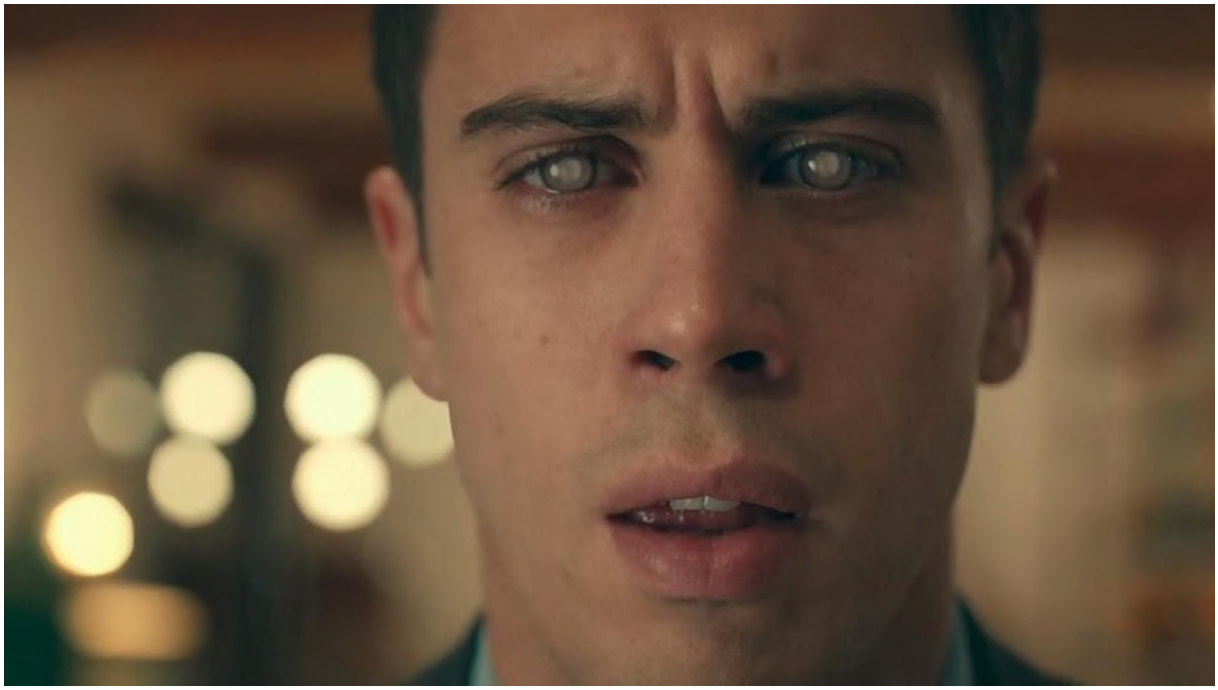
SUMÁRIO

INTRODUÇÃO

Com o avanço da tecnologia, a maior parte dos trabalhos como conhecemos hoje está desaparecendo; seja devido à automatização com robôs ou pela simples falta de necessidade. Ao mesmo tempo, são desenvolvidas novas formas de entretenimento; afinal, sem trabalho o ser humano tende a sofrer com o ócio. O entretenimento é tão antigo quanto a humanidade em si; o homem sempre buscou formas de passar o tempo: por meio de histórias, artes, música, esportes, jogos...E todos esses ainda são os principais meios até hoje, sofrendo, obviamente, alterações em diferentes níveis. Nos últimos anos, no entanto, surgiu uma nova opção que veio para ficar: a realidade virtual.

Mas o que é realidade virtual? Atualmente, associa-se imediamente àqueles óculos com telas embutidas que se prende na cabeça, como na foto abaixo. O que esses aparelhos fazem, basicamente, é projetar nessa tela ambientes gerados em 3D pelo computador e capturar os movimentos feitos pelo usuário; ao isolar a pessoa do mundo real, ela tem a impressão de estar em outro lugar, um lugar gerado virtualmente. No entanto, como disse Jonathan Steuer em seu artigo "*Definindo realidades virtuais: Dimensões determinando telepresença*", não é o ideal que liguemos a realidade virtual a essas tecnologias, pois isso quantifica as maneiras que ela pode ser experenciada; por exemplo, talvez haja, no futuro, algum implante que envia imagens diretamente para o cérebro - o que dispensaria a necessidade desses óculos especiais.

Mulher usando um headset VR da Samsung na SXSW, 2015 < https://goo.gl/tfEVSz>



“Black Mirror”, 2011

Independente de como ela evoluirá, a realidade é que hoje não é uma tecnologia utilizada somente para o entretenimento: já há aplicações em diversos campos como a medicina, arquitetura, pedagogia, esportes, artes...E é neste momento que o designer se torna, mais uma vez, importante: é a sua responsabilidade fazer a ponte entre a técnica e o usuário; é ele que pegará toda essa tecnologia, todas essas possibilidades, e entregará algo que pessoa comum possa entender, que a permitirá tirar o máximo de sua capacidade - é o designer que desenvolverá a experiência.

E o objeto desta dissertação é preparar esse designer para esta responsabilidade; é um guia que mostrará como produzir, do início ao fim, uma cena em realidade virtual. Não se trata de uma tarefa fácil, certamente; são muitos softwares a se dominar, muitos conceitos a se aprender, mas que, após dominar toda essa parte técnica, é extremamente gratificante: é possível criar mundos e experiências jamais antes vistas.

Como exemplo, contudo, voltaremos ao passado: recriaremos uma das grandes obras do passado neste novo meio; ou seja, trata-se de mais um uso da realidade virtual: apresentar uma nova forma de se experenciar as maravilhas feitas pelos nossos antepassados, trazendo-as, também, aos públicos mais jovens. Neste caso, trata-se do conto “Assassinatos na Rua Morgue”, do mestre do macabro Edgar Allan Poe. Vamos recriar o quarto onde se passa a história, com todos os seus detalhes, para que o leitor (ou usuário) possa ver por si mesmo todo aquele clima macabro que aquele grande escritor quis nos mostrar ao contar sua história. Mãos à obra!

PRÉ-PRODUÇÃO

Desde o começo do curso, em praticamente todas as matérias, os professores são unânimes em uma opinião: o quão importante é esboçar, e pensar no trabalho antes de começar a fazê-lo propriamente. Este é um conselho que, vergonhosamente, eu não dava a devida atenção, mas que, com a experiência, vim a perceber sua importância. Afinal, não importa que atividade esteja se fazendo, é quase senso comum de que algo que começa errado dificilmente dará certo no final.

Apesar de os professores não terem dado um nome a esta etapa, na indústria ela é devidamente batizada: pré-produção. É aqui que se decide praticamente todas as questões do projeto, do início ao final. Por isso é tão importante: ao encontrar um problema, é muito mais fácil contorná-lo e adaptar o percurso quando já se sabe aonde está indo. E isso se torna particularmente verdadeiro em um projeto que usa 3D, que possui diversas etapas até ter o render final.

Cada um tem sua maneira de trabalhar, porém há alguns pontos que definitivamente não devem ser ignorados antes de começar um projeto, sobre os quais discorrerei a seguir.

2.1. ESCOLHA DOS SOFTWARES

“Um computador não cria animação computadorizada mais do que um lápis cria animação tradicional. Quem faz animação computadorizada é o artista.” (LASSETER, John)

Esta frase, dita por uma maiores mentes do mundo da animação e da computação gráfica, John Lasseter, que é chefe de criação dos renomados estúdios Pixar e Walt Disney, reflete bem a profissão, não só de animador, mas do designer em geral: apesar de usarmos softwares para as nossas criações, no fim das contas a qualidade do resultado final depende exclusivamente da capacidade do profissional.

No entanto, a evolução tecnológica progride com uma velocidade tão assutadoramente grande que nem mesmo os mais brilhantes cientistas do passado puderam prever. E cada avanço torna os trabalhos mais fáceis, permitindo focar no que realmente importa. É por esse motivo que se diz comumente que designer é uma profissão em que não pode se parar de estudar. Além de ser preciso acompanhar as tendências, saber usar os softwares mais sofisticados retira uma boa parte do parte repetitiva, o que, por consequência, permite que se passe mais tempo aumentando a qualidade.

Nesse sentido, é fácil se perder ao olhar a quantidade de softwares 3D disponíveis no mercado. Ao analisar com mais cuidado, no entanto, vemos que muitos deles tem usos bem específicos, o que nos permite separá-los por categoria.

A primeira categoria, e indiscutivelmente uma das mais importantes, é a chamada *3D package*. Se refere aos programas "faz-tudo", que permitem criar objetos 3D e levá-los até o final da etapa, o render, passando por todas as outras etapas, como *rigging* e animação.
Ou seja, trata-se de um software obrigatório, que contém várias das ferramentas que você vai precisar. É aqui também onde se encontra a maior concorrência. São vários nomes: Maya, 3ds Max, Blender, Cinema 4d, Houdini, Modo...Sendo que os dois primeiros são os mais usados na indústria, e os únicos que este autor conhece a fundo.

Apesar de o Maya ser um excelente software, e normalmente a minha primeira escolha, neste caso decidi usar o 3ds Max, que usualmente é usado para fazer visualizações de arquitetura, o que se aproxima mais do que estamos criando aqui. E isso se dá pelo fato dele ser bem intuitivo para se trabalhar com medidas exatas, o que definitivamente queremos.

Após a criação do objeto 3D e a sua exportação do 3ds Max, o próximo passo é tão importante quanto o primeiro, pois é aqui que definimos como será o visual do *asset*. Trata-se da texturização, e é uma etapa mais artística do que técnica em si, pois é aqui que pintamos e exportamos os mapas que serão usados no *shader*, cálculo que define como o objeto reage a luz e é renderizado.

Também são encontradas várias opções de softwares de texturização, mas neste caso a escolha é mais fácil. O Photoshop foi, por muito tempo, a principal escolha para este trabalho; muito mais pelo fato de que não havia outras opções. No entanto, ele apresenta grandes disvantagens: como é um software primariamente 2D, os artistas tinham que pintar os mapas diretamente na UV (que é a versão planificada do objeto); ou seja,

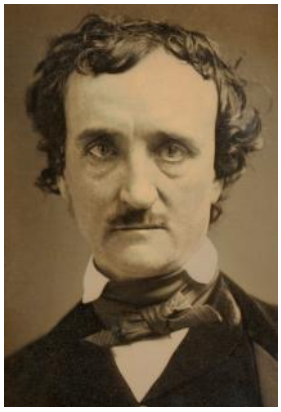
não era possível ver o objeto no espaço 3D enquanto se pinta. Além disso, o ato de texturizar não se trata de criar apenas uma camada de cor; é necessário criar outros mapas, como o mapa de *roughness*, que define se a superfície é áspera ou lisa. E com o Photoshop temos que criar todos esses mapas separadamente sem saber como ficará o resultado final, o que resulta em uma grande quantidade de tempo perdida, pela necessidade de ficar indo e voltando da *engine*, para ajustar as texturas.

Por isso minha escolha foi um software chamado Substance Painter. Criado especificamente para texturizar modelos que serão usados em ambientes *real time,* ele tem tudo o que poderíamos esperar para facilitar o trabalho e mais: além de ver o modelo em 3D, sua *viewport* age como uma *game engine*, o que nos permite ver praticamente como será o resultado final enquanto ainda estamos texturizando; ele também nos permite pintar todos os canais (cor, *roughness*, normal) *ao mesmo tempo,* o que garante a uniformidade entre eles; conta com vários materiais já prontos para uso, o que reduz drasticamente o tempo de texturização, o que se torna extremamente importante quando é preciso texturizar vários objetos; além disso, em sua última versão foi introduzido um *live link* entre o Substance Painter e a Unreal Engine, que atualiza automaticamente na *game engine* as texturas geradas, sem necessidade de exportá-las novamente toda vez que for feita uma mudança, o que, mais uma vez, nos salva bastante tempo. Este software não só foi uma evolução importantíssima na indústria, mas também tornou o processo de criação de texturas muito mais prazeroso.

Finalmente, e não menos importante, é preciso fazer a escolha do motor gráfico que irá renderizar em tempo real a nossa cena, também comumente chamado de *game engine,* por ser normalmente utilizado para fazer jogos. Aqui, apesar de haver uma certa concorrência, percebemos que dois softwares dividem o mercado, sendo os outros muito menos utilizados. São eles a Unity e a Unreal Engine 4. Ambos são excelentes no que fazem, entregando resultados bem similares, tem comunidades ativas e desenvolvimento constante, o que faz com que essa seja uma escolha mais pessoal. Após testar ambos, decidi por utilizar a Unreal Engine, pois percebi que conseguia um resultado visual melhor mais rápido do que na Unity, e também porque a interface me pareceu bem mais agradável e simples de utilizar, o que também deve ser levado em consideração, pois o desenvolvedor passará boa parte do tempo lidando com esta interface.

2.2. REFERÊNCIAS

Após decidir o que será feito e reunir as ferramentas necessárias, chega a hora de começar a definir o visual e criar o conceito sobre o qual designer irá se basear a começar o processo de criação propriamente dito. Para isso, não importa o que se esteja fazendo, é extremamente importante ver o que já foi feito, não para copiar, como algumas pessoas acreditam, mas para garantir que o seu trabalho não foge muito do esperado, caso essa não seja a sua intenção.

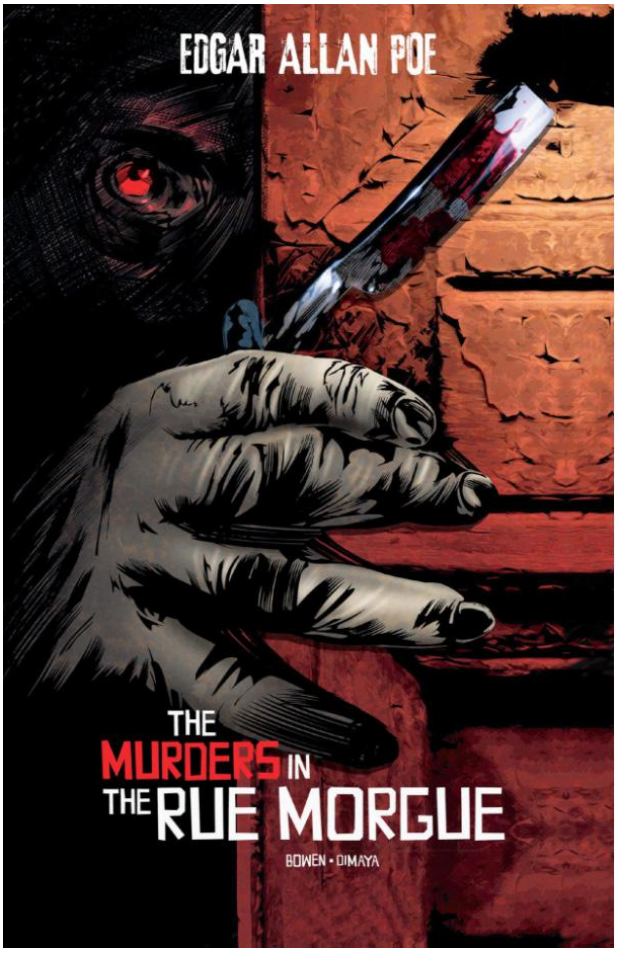


Como já dito anteriormente, este projeto é baseado na obra "Os Assassinatos da Rua Morgue", de Edgar Allan Poe, autor que viveu durante a primeira metade do século XIX e é extremamente famoso por seus contos e poemas com temas de mistério e macabro. É também considerado o pai do gênero de ficção investigativa, no qual se encaixa este conto de que falamos.

Na história, que se passa no século XIX em Paris, uma mãe e sua filha são assassinadas de forma brutal no apartamento onde viviam sozinhas. Obviamente, a primeira fonte sobre a qual nos devemos basear é a obra original. O problema é que, apesar de Poe descrever minuciosamente o quarto onde a história se passa, essa referência é escrita, e não temos nada visual para nos guiar. De acordo com o autor, era um quarto simples, com uma cama para as duas pessoas que nele viviam, duas janelas, uma lareira e chaminé que saia para fora do quarto e que estava completamente bagunçado.

A segunda fonte principal de referência é novamente o próprio conto, só que desta vez em forma de *graphic novel*; ou seja, temos ilustrações. O quadrinho, que foi ilustrado por Emerson Dimaya, nós dá um pouco mais de ajuda na imaginação, porém, como toda ilustração, é influenciado pelo estilo de quem desenha, o que neste caso se trata se desenhos mais simples, com cores sólidas e pouco detalhamento.

Há também um filme chamado "O Corvo", produzido nos Estados Unidos em 2012, que conta, com uma grande liberdade criativa, os últimos dias de vida de Poe. Na história, são cometidos assassinatos em condições quase idênticas as descritas por Edgar em alguns de seus contos, e o escritor é chamado para ajudar a resolver estes crimes. O primeiro assassinato é baseado em "Os Assassinatos da Rua Morgue", e, apesar de ser curta a cena, nos serve como referência também.



"The Murders in the Rue Morgue", BOWEN, Carl, DIMAYA, Emerson, 2014

"O Corvo", 2012

Após juntarmos referências originais, partimos para fontes externas. Pela descrição do conto, podemos nos situar historicamente e começar a filtrar a nossa busca. Busquei aqui então filmes e seriados situados em ambientes urbanos do século XIX. Deste período da história é muito fácil encontrar histórias sobre a realeza e as castas mais altas da sociedade da época, o que não necessariamente nos interessa, pois havia muita diferença entre as habitações de pessoas ricas e pobres. Neste sentido, a primeira série que me chamou atenção foi Penny Dreadful, que apesar de ter um tema sobrenatural, se passa na Inglaterra Victoriana e conta com personagens “comuns”.

Diversos outros filmes que se passam na mesma época também foram usados como inspiração, mais como referência de design de interiores e iluminação, como “Orgulho e Preconceito”, “Do Inferno” e “Sweeney Todd: O Barbeiro Demoníaco da Rua Fleet”.

PRODUÇÃO

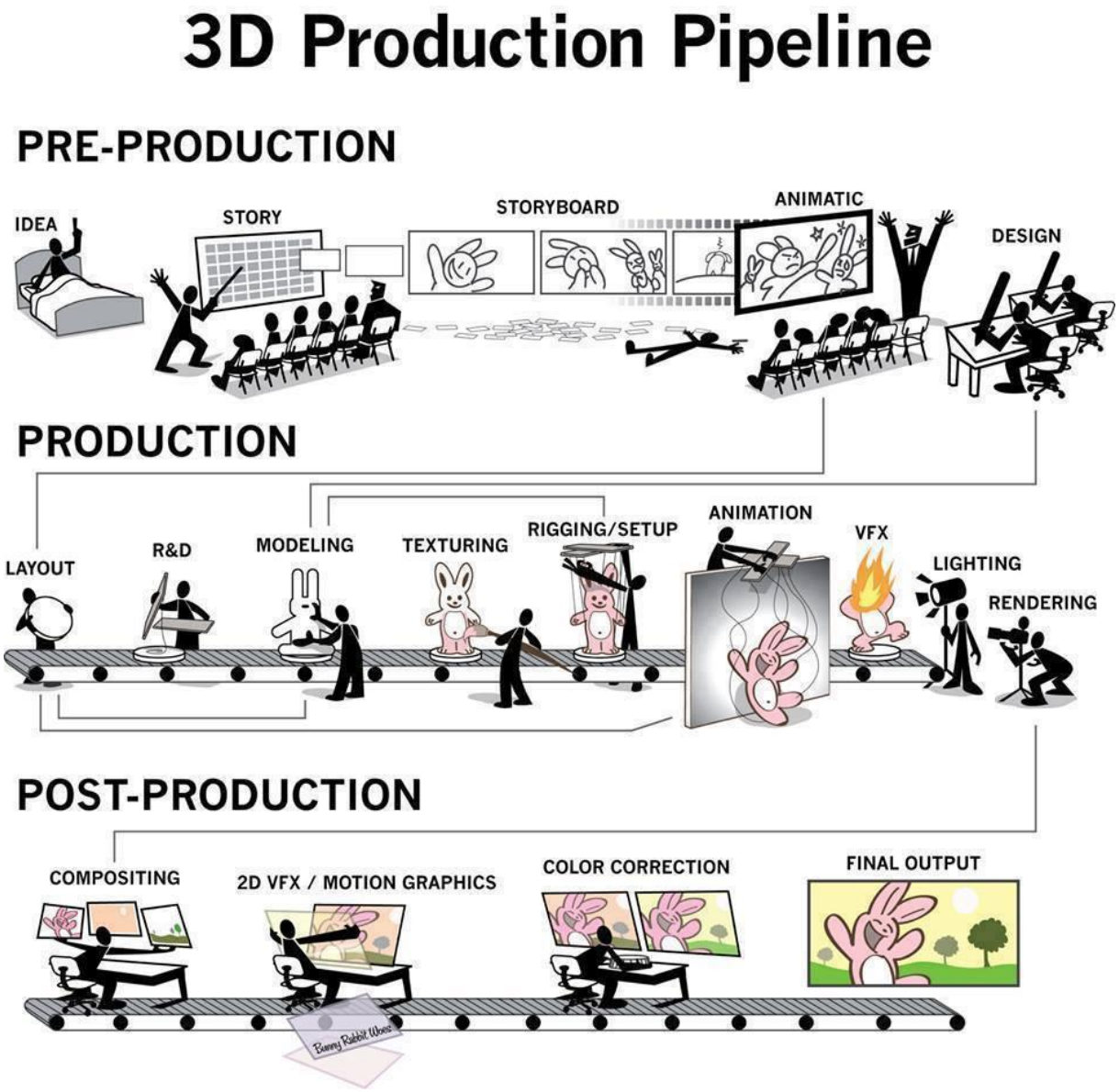


Gráfico mostrando o processo de produção de uma animação 3d, < https://goo.gl/QAT77E>

Tendo em mãos o conceito, as referências e os softwares necessários, é hora de entrar na fase de produção propriamente dita. Em uma produção 3d tradicional, como em uma animação, esta f ase é composta pelas seguintes etapas (sujeitas a pequenas alterações dependendo do estúdio): modelagem, texturização, rigging, animação, efeitos, iluminação e render (como exemplificado na imagem acima). Como no caso deste trabalho será produzido um cenário estático, ou seja, não haverá animação, pode-se pular as etapas de rigging e animação. A seguir são explicadas cada uma delas, em ordem de execução.

3.1. MODELAGEM

Algumas das etapas da pipeline podem ser feitas em ordens diferenciadas; ou seja, é possível riggar o modelo antes de texturizá-lo, ou iluminar uma cena sem ter as animações prontas, mas não é possível concluir nenhuma delas sem ter os modelos 3D em si. Por isso, é comum começar a produção pela modelagem. Para esta parte, o software 3ds Max será suficiente.

Como já dito anteriormente, produzir conteúdo para realidade virtual não é tão diferente de se produzir um jogo; mais precisamente, o processo é exatamente o mesmo, o que muda sendo o uso do resultado final. Qualquer um que já tenha jogado algum game, principalmente os que foram lançados até a década passada, e assistido alguma animação, em especial aquelas dos grandes estúdios, como Pixar e DreamWorks, deve ter percebido uma enorme diferença de qualidade entre um e outro. É uma questão de frustração para os gamers até hoje: é comum vê-los reclamando que "o jogo não parece em nada com o trailer". Mas por que isso acontece? O trailer, também conhecido como cinemática neste meio, nada mais é do que um curta-metragem. Como toda animação ou filme, ou seja, mídias que não possuem interatividade, ele é *pré-renderizado*: ou seja, é possível usar a qualidade máxima disponível, se preocupando muito pouco com detalhes como número de polígonos, tamanho das texturas ou qualidade das sombras, detalhes esses que demandam um alto poder computacional. A única coisa que acontece quando se deixa uma cena mais complexa e com maior detalhamento é o aumento no tempo para se renderizar um frame. Um frame em animações de grande porte pode levar horas para ficar pronto. Porém, isso não é possível em jogos. Devido à interatividade, não se sabe para onde o jogador vai olhar, andar, enfim, que ação ele irá tomar, e por isso o game precisa ser renderizado em tempo real; ou seja, o frame precisa ficar pronto em frações de segundo. E para isso acontecer é preciso economizar poder computacional onde for possível, caso contrário ficará "injogável". É importante notar, no entanto, que isso é uma limitação temporária: há 20 anos atrás, um personagem principal de um jogo, como o famoso Crash Bandicoot do PlayStation, não costumava passar de 1,000 polígonos (Crash, por exemplo tinha 532). Hoje em dia é comum termos personagens com dezenas de milhares de polígonos. Com o avanço da tecnologia, é natural que os computadores possam lidar com cenas cada vez mais complexas, até chegarmos a um ponto em que não há real diferença entre é uma cena pré-renderizada e uma real time.

Comparativo entre Crash Bandicoot (1996) e Crash Bandicoot - N'Sane Trilogy (2017), < https://goo.gl/snYYMm>

Atualmente, no entanto, ainda é preciso tomar alguns cuidados a se tomar ao criar elementos para cenários renderizados em tempo real, começando pela modelagem. Apesar de um computador mediano hoje em dia ser capaz de lidar com cenas com milhões de triângulos, um celular ainda não é capaz de fazer o mesmo, e a minha intenção é nivelar por baixo, para que a cena possa ser lida pela maior quantidade possível de dispositivos. Portanto, o ideal é que cada objeto tenha a menor quantidade possível de polígonos. Ao mesmo tempo, queremos criar um ambiente fotorrealista e modelos com muitas poucas faces tiram essa sensação de realismo, como no exemplo ao lado. Uma boa regra é deixar as partes que tem curvatura com uma maior densidade, ao contrário das áreas retangulares.

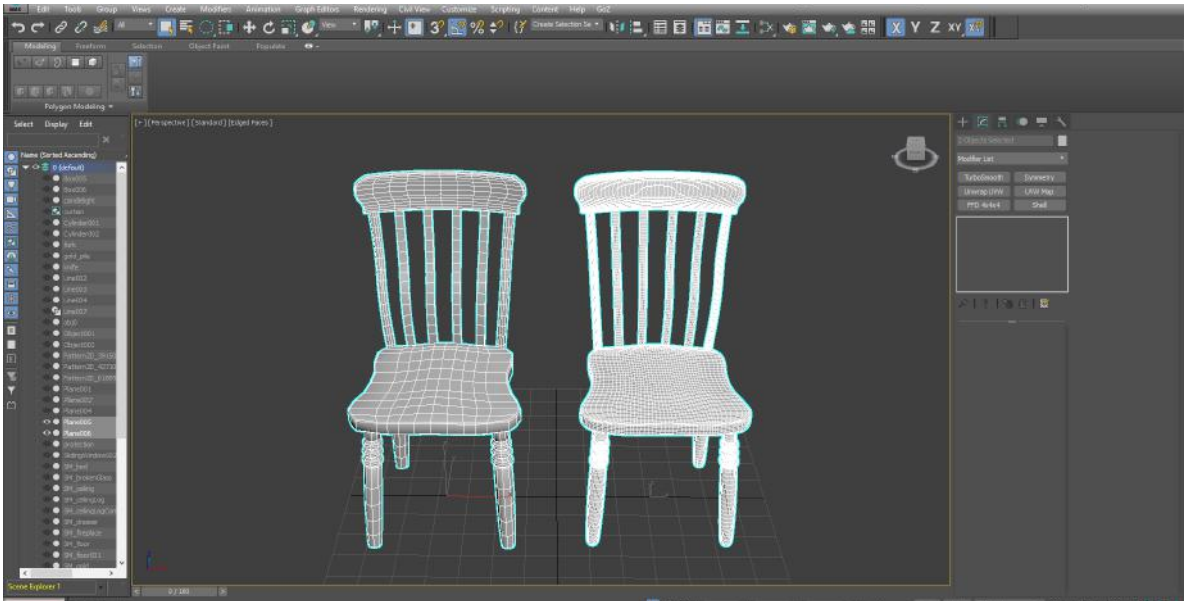
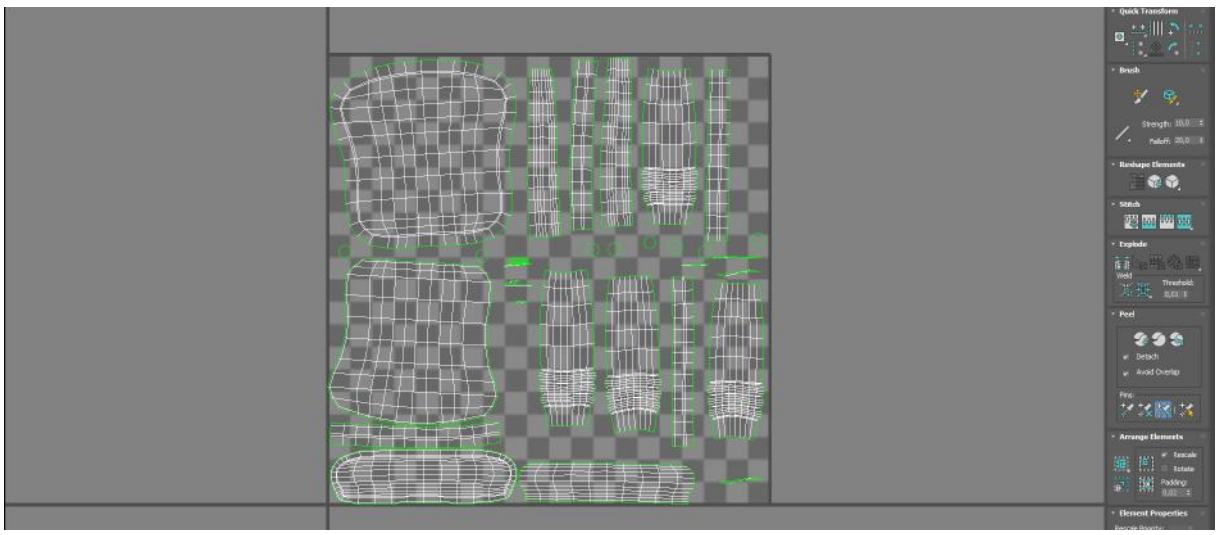


Imagem comparando diferença entre um modelo low-poly *e um* high-poly *(arquivo pessoal)*

Não nos preocupamos com detalhes da superfície: em sua maioria, estes serão lidados pelo mapa de normal. Este mapa, que é extremamente importante, tem a função de simular deformações no objeto, sem modificar a geometria em si. Entraremos em mais detalhes na parte da texturização, que é quando o geramos.

Após ter o modelado o objeto, é preciso fazer o mapeamento UV do mesmo, o que é chamado corriqueiramente de “abrir a UV”. Este processo, apesar de estar incluido na fase da modelagem, tem mais a ver com a fase da texturização do que da modelagem em si; porém, é uma prática comum sempre entregar o modelo com o devido mapeamento UV. Abrir a UV significa basicamente que vamos planificar o nosso objeto, que está em três dimensões, para que receber arquivos de imagem (que estão no espaço 2D) contendo as informações da textura. É como pegar pegar um cubo, cortá-lo em duas arestas e planificá-lo, para que possamos ver todas as faces ao mesmo tempo. Esta ação é necessária pois não há como pintar diretamente no modelo; para ser mais preciso, tal tecnologia até existe - se chama Ptex e foi desenvolvida pela Pixar -, mas ainda não há suporte para ela nos renderizadores em tempo real atuais.

O mapeamento UV é também um processo que deve ser feito com cuidado, embora isso se aplique a todos os tipos de modelos, não somente aqueles feitos para games. Em geral, é preciso ter cuidado com duas coisas: a primeira é criar a menor distorção possível. Durante a planificação, é preciso se lembrar de usar uma ferramenta que normalmente se chama “relaxar”, que vai deixar os polígonos igualmente espaçados, o que evitará distorções na UV e, consequentemente, na textura. A segunda é na quantidade de ilhas (as partes que são separadas ao fazer os cortes). Quanto mais ilhas a UV possuir, normalmente mais “seams” (os cortes) são visíveis na textura, principalmente ao se texturizar em um software 2d, como o Photoshop. Porém, ao usar o Substance Painter, pintamos diretamente no objeto 3d, o que significa que esse problema é eliminado.



Exemplo de um modelo com UVs abertas (arquivo pessoal)

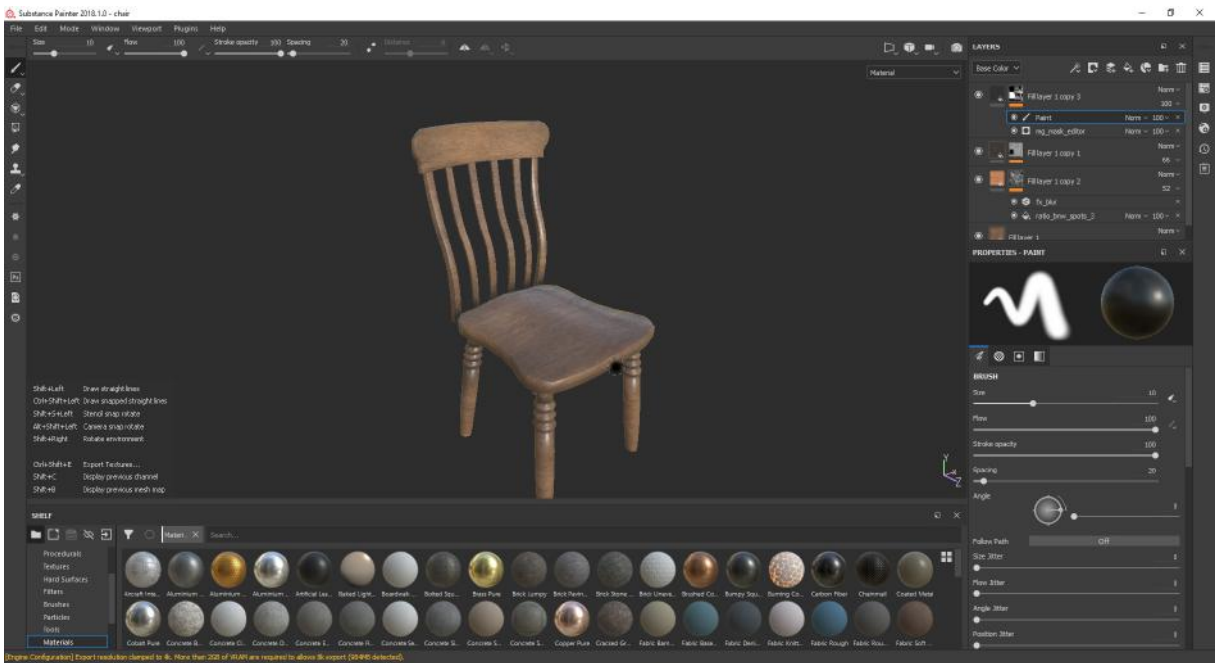
Aqui surge um problema: fazer o mapeamento UV é um processo monótono e, caso se queira fazê-lo perfeitamente, requer uma boa quantidade de tempo. Quando se trabalha com diversos objetos, como no caso deste trabalho, pode acabar se perdendo tempo valioso em uma etapa que não influencia tão diretamente no resultado final. É por este motivo que decidi usar, sempre que possível, o mapeamento automático do 3ds Max, que literalmente abre as UVs, sem distorção, com somente um clique. Obviamente, o resultado fica longe do perfeito, criando um mapa pouco organizado e com várias pequenas ilhas, mas que, como já dito antes, não é o problema quando texturizamos usando o Substance Painter. Com um pouco de trabalho, como rotacionar todas ilhas para que fiquem na mesma direção e eliminar cortes desnecessários, o resultado fica perfeitamente aceitável para o nosso propósito.

Com isso, o modelo está finalmente pronto para a exportação. Ao exportar, é muito importante nomear devidamente o objeto, para que seja fácil de achá-lo no projeto depois, e para que seja intuitivo, se, por acaso, alguma outra pessoa for mexer no arquivo. Há um sistema de nomeação que é usado frequentemente por usuários da Unreal Engine, que inclusive está explicado em sua documentação. Para a cama aqui representada, por exemplo, por ser um objeto estático, ficaria desta maneira: *SM_cama* (SM significando *static mesh*). Após isso, só resta escolher o formato de saída. São dois os formatos mais comuns: .OBJ e .FBX. Em minha experiência, o segundo costuma menos problemas, por isso será o escolhido.

3.2 TEXTURIZAÇÃO

Chega a hora de dar vida aos nossos modelos usando o Substance Painter, que será o único software utilizado nesta fase. Ele só lida com um objeto por vez, por isso precisamos criar uma nova cena para cada asset que for ser texturizado. Na janela de criação, como podemos ver na imagem abaixo, após importar o arquivo, é possível escolher um *preset*, o que é extremamente útil, pois ele já adapta o shader e prepara as texturas para exportação de acordo com o workflow desejado. No nosso caso, escolhemos o preset Unreal Engine 4.

Aqui é de suma importância, como em todas as outras etapas, pensar no que será feito antes de começar o trabalho em si. Por exemplo, vamos texturizar objetos pertencentes a uma residência de classe média/baixa do início do século XIX. Usaremos, portanto, materiais como madeira envelhecida, tijolos, metais como ferro e bronze desgastados, etc. Mas por que é preciso saber disso antes?



Cadeira texturizada no Substance Painter (arquivo pessoal)

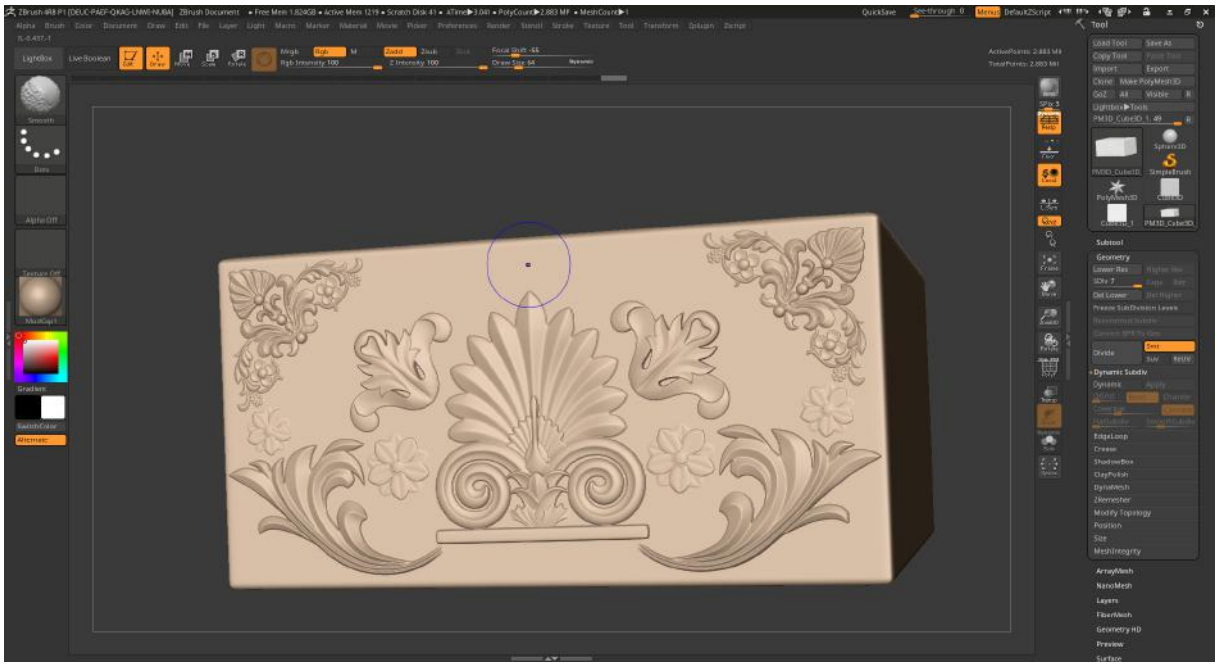
Porque o Substance Painter, apesar de permitir a pintura tradicional, se destaca pelo uso de procedurais: materiais já prontos, ou seja, que já possuem um visual pré-definido, mas que são customizáveis. Por exemplo, ao invés de criarmos uma madeira do zero, o que daria um trabalho enorme, podemos pegar o material já pronto e adaptá-lo a nossas necessidades, como mudar a cor, a quantidade de tábuas e a idade da madeira. É possível criar os seus próprios procedurais utilizando um software “parente” do Substance Painter, o Substance Designer, mas isto esta fora do escopo deste trabalho. A Allegorithmic, empresa dona de ambos os softwares, possui dois domínios onde os

usuários podem baixar esses materiais: o Substance Share, em que os próprios artistas compartilham de graça com a comunidade, e o Substance Source, que são procedurais de alta qualidade feitos pela própria Allegorithmic e por estúdios especializados. Utilizaremos ambos os websites para termos o menor trabalho manual possível.

Cama texturizada no Substance Painter (arquivo pessoal)

Ao texturizar essa cama, por exemplo, são dois os principais materiais de base que a compoem: a madeira, para a cama em si, e o linho, para o colchão e os travesseiros. Em ambos os casos, começamos com procedurais, ajustamos um pouco os seus parâmetros e obtemos logo de cara um resultado satisfatório, porém genérico. É possível reduzir bastante o trabalho manual, mas caso queiramos algo fora do comum, não podemos evitá-lo, pois é isso que dá a diferença no final. Os eventos se passam durante/logo após um período artístico conhecido como Rococó, um desdobramento do Barroco, que influenciou principalmente a arquitetura, decoração de interiores e ornamentação de objetos. Seu estilo se caracterizava pelo rebuscamento, uso de linhas curvas e cores claras, misturando-se ao dourado e ao prateado. Uma de suas características mais marcantes era o detalhamento, como entalhes desenhados na madeira e no metal, e com certeza queremos colocar isso em nossos objetos. Normalmente este tipo de detalhamento, que modifica a malha em si, é feito em um programa de escultura, como o ZBrush.

Funciona da seguinte maneira: após exportar o modelo *low-poly* (baixa resolução), o importamos no ZBrush, onde subdividimos a malha diversas vezes até que fique em alta resolução, para que possamos esculpir pequenos detalhes. Após a escultura, exporta-se o mapa de normal, que simulará esses detalhes na textura, sem a necessidade de usarmos um modelo super denso. A diferença do Substance Painter é que nele pode-se pintar *diretamente no normal map*, eliminando esta etapa do meio. Obviamente, o mapa de normal não produz um resultado tão bom quanto uma malha de fato em alta resolução, mas servirá suficientemente bem para o nosso propósito.

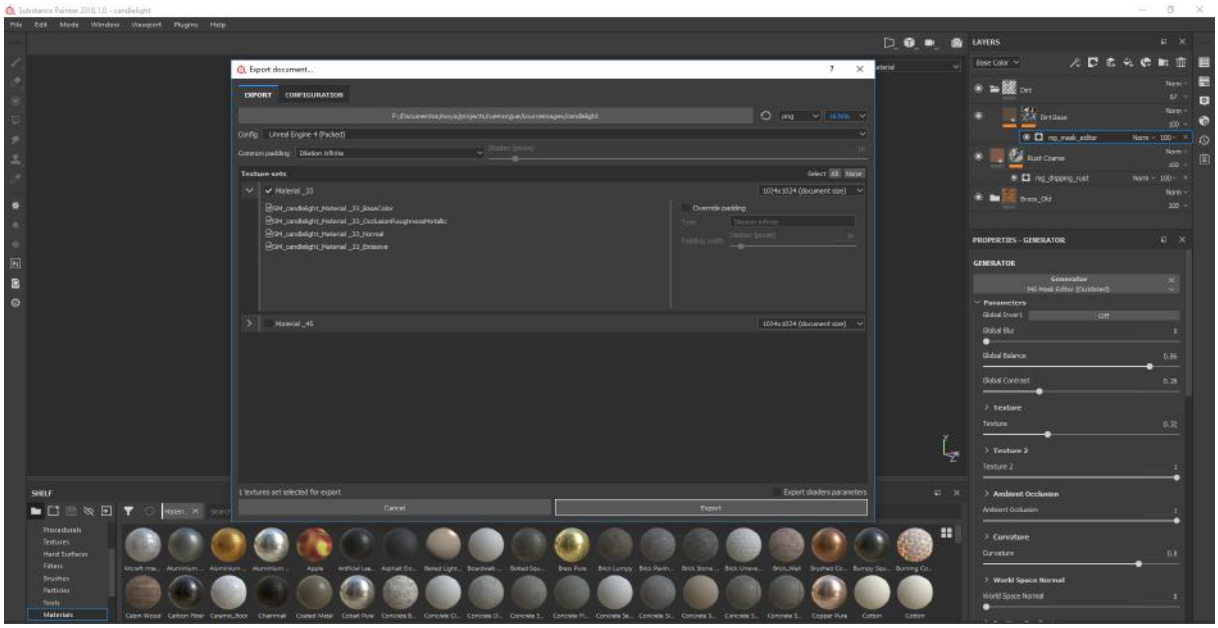


Exemplo de escultura de detalhes no Zbrush (arquivo pessoal)

Após os detalhes na superfície, parte-se para a texturização em si. Queremos criar uma atmosfera mais assustadora e “suja”, de certa maneira, que condiz com o clima da história, por isso adicionamos uma camada de poeira em quase todos os objetos, ferrugem em metais, etc. Fora isso, é preciso analisar caso a caso. De acordo com a descrição de Edgar Allan Poe, havia sangue espalhado por todos os lados. Na cama, por exemplo, faz sentido espalhar sangue sobre o colchão e travesseiros, além do fato de o vermelho sobre o branco do tecido produz uma imagem forte e impactante.

Alguns dos objetos texturizados no Substance Painter (arquivo pessoal)

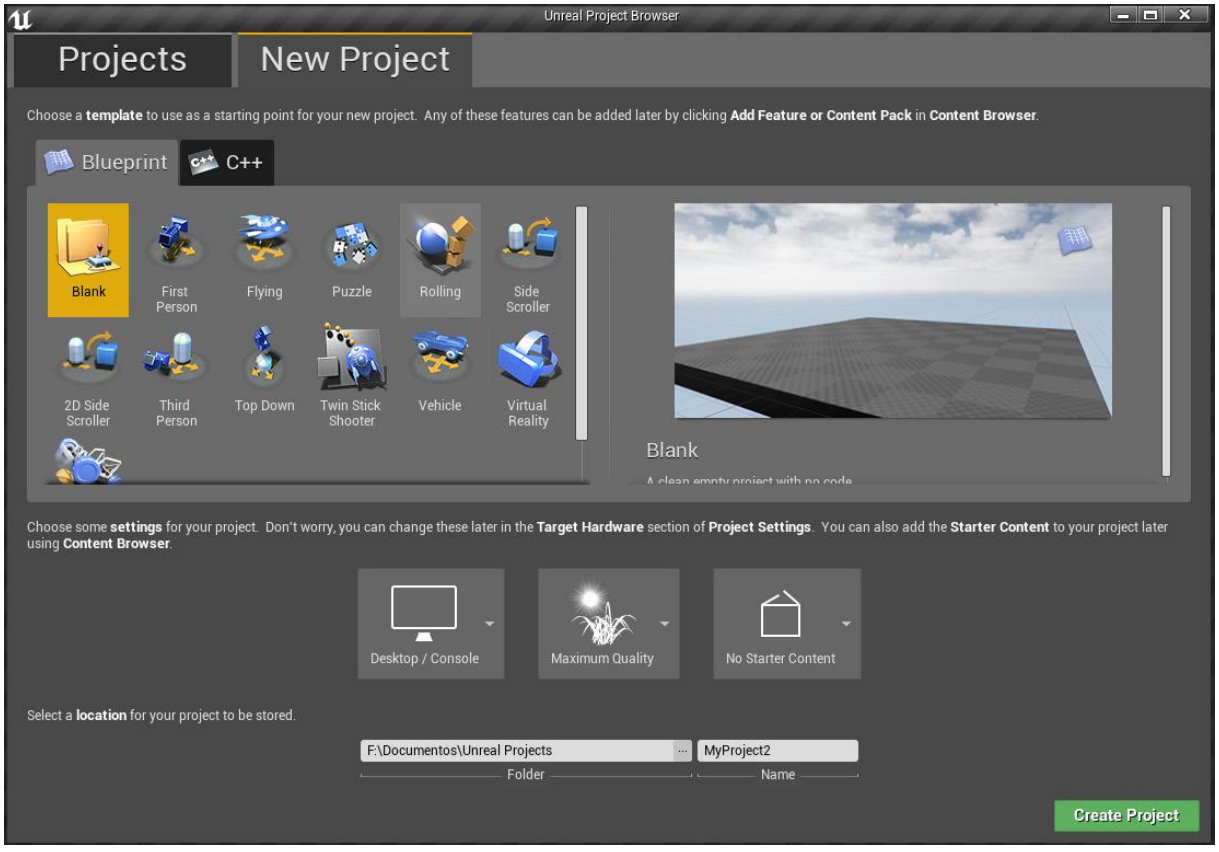
Ao finalizar a pintura dos mapas, chega a hora de preparar para a exportação. Como no início do projeto foi escolhido o preset Unreal Engine 4, não há muito o que fazer em termos de customização, pois o software já cuida desta parte. Somente é preciso definir duas coisas: o formato dos arquivos e o tamanho. Para nosso uso, .JPEG será suficiente. Já em relação ao tamanho das texturas, é preciso analisar com cuidado o caso de cada modelo. Idealmente, exportaríamos tudo em 4096x4096px, o que produz uma qualidade excelente, mas ficam muito pesadas e nem sempre é necessário ter toda essa resolução em objeto pequeno e que será visto de longe, como o candelabro, por exemplo - para este caso 1024x1024 já é satisfatório.



Exemplo da tela de exportação de texturas do Substance Painter (arquivo pessoal)

3.3 MONTANDO A CENA NA UNREAL

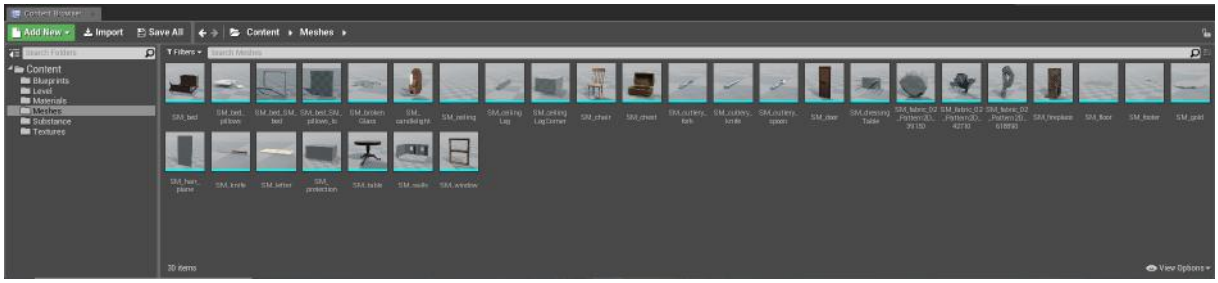
Com os modelos e as texturas prontos e separados, é hora de trazer tudo para a Unreal Engine, que é onde se desenrolará o resto do processo. O primeiro passo, como sempre, é criar um projeto novo, e aqui somos presenteados com algumas opções como: modo de programação por blueprints ou por C++; qualidade alta, média ou baixa; projeto voltado para PC/console ou smartphones (mobile); e finalmente com ou sem conteúdo inicial. Caso seja necessária alguma programação no projeto, para um designer que não tenha expericiência anterior com a linguagem C++, o modo por blueprints é espetacular, pois ele permite uma programação visual, por nodes, sem a necessidade de escrever código. Por isso, escolhemos um *projeto em branco, guiado por blueprints*. As escolhas feitas a seguir dessa são reversíveis depois, portanto iremos com as seguintes configurações a princípio: PC/console; alta qualidade; sem conteúdo inicial (para que o projeto fique leve e só contenha aquilo de que realmente necessitamos).



Tela de criação de novo projeto na Unreal Engine 4 (arquivo pessoal)

Com o projeto criando, pode-se começar a importar os assets. Para manter tudo organizado, é de praxe criar pastas para separar os arquivos de acordo com o seu tipo. Criamos as pastas: Meshes; Texturas; Materiais; Level; Blueprints. Nas duas primeiras importamos, respectivamente, os modelos e as texturas criados anteriormente. A pasta Level (também chamada de Mapa por alguns) se refere ao nível do jogo que estamos criando; como nesse caso só temos um ambiente, ou seja, somente um "nível", o salvaremos nesta pasta - mas é possível criar outros no mesmo projeto. Na pasta Blueprints é onde ficarão as blueprints de programação, caso haja alguma.

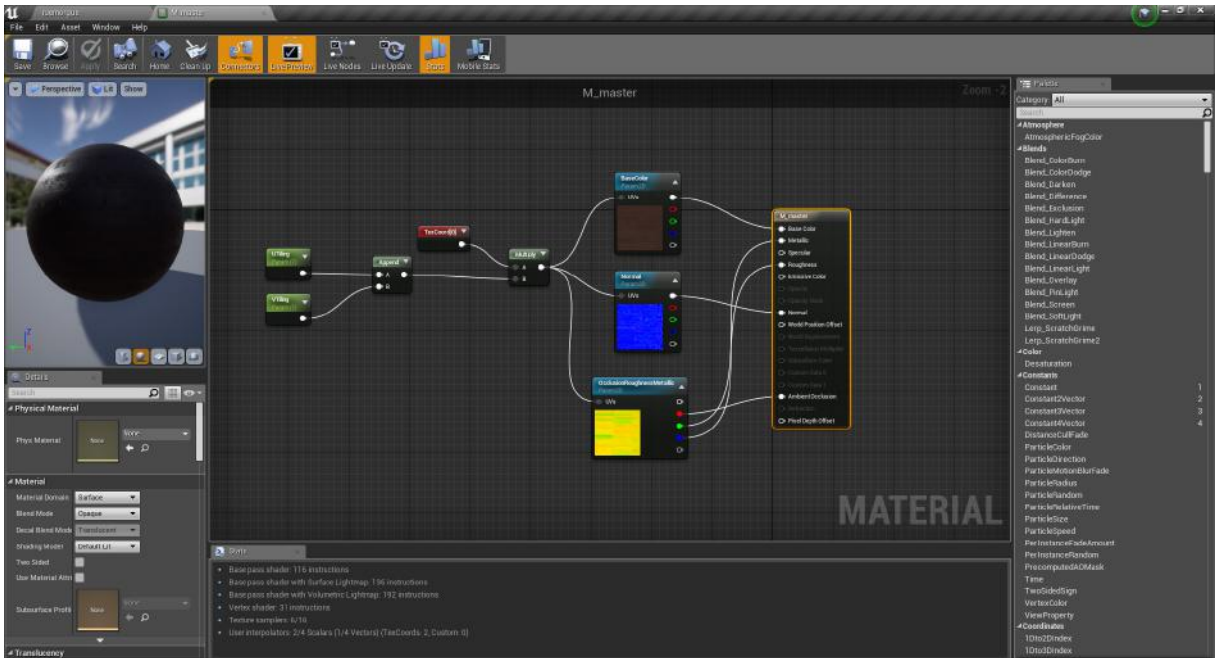
Além da organização por pastas, é extremamente importante nomear corretamente todos os elementos do projeto, para que seja seja fácil achá-los dentro do projeto. Entre os desenvolvedores que usam a Unreal Engine, existe uma convenção para nomenclatura que é universalmente aceita: adicionar o prefixo SM_ (que significa *Static Mesh*) para modelos estáticos; T_ (*Texture*) para texturas; M_ e MI_ (*Material* e *Material Instance*) para materiais e inst âncias de materiais, respectivamente; BP_ (*Blueprint*) para *Blueprints*; e por aí vai. Uma lista completa pode ser achada em: https://wiki.unrealengine.com/Assets_Naming_Convention.



Exemplo de organização de projeto (arquivo pessoal)

Os Materiais demandam uma atenção especial. Também conhecidos como *shaders*, eles são a ponte entre o modelo 3D e as suas texturas: eles definem como estas texturas responderão a luz, ou seja, como a superfície do objeto será representada na hora do render - por este motivo cada renderizador possui os seus próprios shaders. Na Unreal existem dois tipos: *Material* e *Material Instance*. O primeiro é o shader como conhecemos. Já o segundo é um derivado do shader: é um material pré-calculado que permite a mudança de determinados parâmetros. E saber isto é extremamente importante na otimização de projetos: o material é mais custoso para se calcular em tempo real, enquanto a sua instância tem um custo quase zero. O Substance Painter nos dá três imagens quando exportamos as texturas: Diffuse, AmbientOcclusionRoughnessMetallic e Normal. Essas três imagens são suficientes para se montar um shader na Unreal - na verdade, se tratam de 5 texturas, só que três delas (AmbientOcclusion, Roughness e Metallic) são em escala de cinza e seria um desperdício colocá-las isoladas, por isso o

Substance põe as três em uma imagem só, nos canais R, G e B, respectivamente. Em outras palavras, ao trocá-las podemos transformar o material da chaminé no material da porta, por exemplo. Por isso podemos criar um material-mestre, com três parâmetros indefinidos no lugar das três texturas. Após isso, criamos somente uma instância para cada objeto, o que nos permite um gasto mínimo de poder computacional.



Tela de criação do shader (arquivo pessoal)

Uma dificuldade surge, no entanto, ao se criar o shader da janela: este objeto é composto por dois materiais com naturezas diferentes, um opaco (madeira) e um translúcido (vidro). Transparência é uma propriedade bem difícil de se calcular em tempo real, e até hoje é difícil conseguir um resultado satisfatório. A Unreal lida com essa questão madeira da seguinte maneira: para se usar a translucência, é preciso mudar o *shading mode* do shader para *Translucency*, o que essencialmente o transforma em outro tipo de material. Isso nos cria um problema: mesmo que criemos uma máscara de opacidade para dizer ao software o que é vidro e o que não é, ele não representa corretamente a madeira ou o vidro, como visto na imagem abaixo, dependendo dos parâmetros que são alterados. Para solucionar esta dificuldade, é preciso separar, dentro do 3dsMax o objeto em duas IDs diferentes (para a madeira e para o vidro) e já na Unreal criar dois materiais separados, para depois aplicá-los a cada uma dessas IDs.

Com todos os objetos em mãos e seus respectivos materiais criados, é hora de montar a cena. Para adicionar um asset basta arrastá-lo para a viewport; é preciso, no entanto, ter cuidado com o posicionamento inicial: ao importar, ele ficará na posição onde você o deixou, mas queremos que ele fique no centro do mundo (0,0,0). Para isso basta zerar esses valores no painel de transformação do objeto. Então é só colocá-los em seu devido posicionamento. A descrição de Poe nos ajudará com esse processo. Na história, ele faz as seguintes afirmações:

- cadeiras e mesas quebradas por todo o lado;
- só uma cama, e dela tudo havia sido puxado e atirado ao chão;
- sangue por todo o lado: no chão, na cama, nas paredes;
- uma faca bem afiada coberta com sangue no chão;
- em frente à lareira havia uma mecha de cabelos prateados, também ensanguentados;
- no chão havia quatro moedas de ouro, um brinco, vários objetos feitos de prata, e duas bolsas contendo grande quantidades de moedas de ouro;
- roupas haviam sido jogadas pelo quarto;
- debaixo das cobertas, uma pequena caixa aberta contendo somente algumas cartas e papéis velhos;
- duas janelas; uma delas facilmente vista, e a outra meio escondida atrás da cama;

Obviamente, não é preciso seguir à risca tudo isso - na verdade, nem podemos, pois como a interatividade é limitada, precisamos pensar no que a pessoa estará vendo. Por exemplo, o quarto certamente deverá ficar bagunçado, mas não a ponto de que não seja possível encontrar certos objetos, como a faca, a caixa, os cabelos e os sacos de ouro, que são essenciais para entender o crime.

3.4 ILUMINAÇÃO

A próxima é etapa é iluminar o cenário; afinal, assim como na vida real, sem luz ficamos na escuridão. A iluminação, como aqueles um pouco familiarizados com cinema e fotografia sabem, é a principal responsável por definir que tipo de atmosfera terá o ambiente, e que sensações ele passará. Como já dito antes, queremos criar um cenário com um visual mais macabro, assustador; para isso é fundamental que o ambiente seja mais escuro - o ser humano tem medo não da escuridão, mas no que nela se esconde. Por isso, é fundamental que nossas luzes não sejam muito reveladoras, mas ao mesmo tempo, por questão de experiência do usuário, não pode ser muito difícil de se avistar os objetos da cena.

Na Unreal Engine, e em praticamente todos os renderizadores real-time, existem três tipos principais de luz: point light, spotlight e directional light. Como o próprio nome já indica, a primeira se trata de um ponto de luz, que ilumina para todos os lados; a segunda funciona como uma “lanterna”; já a terceira simula algo como o sol, com raios vindo de uma distância infinita. E cara uma dessas luzes tem três opções diferentes: estática, dinâmica e móvel. A primeira, como o próprio nome diz, após o início do jogo é impossível movê-la; além disso, ela não calcula sombras em tempo real. Mas para que serve então? Ela tem a grande vantagem de não ter quase nenhum custo computacional, pois o que ela faz é pré-renderizar sombras em objetos estáticos, criando mapas de luz (*lightmaps*) em um processo chamado *bake* de luz - o que não é muito diferente do que um renderizador tradicional *ray trace* faz. A luz móvel é o completo oposto: ela é completamente interativa após o início do jogo, sendo possível movê-la de lugar e mudar todos os seus parâmetros, como intensidade e cor; no entanto, ela não faz o bake de luz e o seu custo computacional é o mais alto de todos, e a qualidade de sombras geradas em tempo real ainda é baixa. A luz dinâmica é uma mistura das outras duas: ela pré-renderiza sombras e também as cria em tempo real; entretanto, não é possível mudá-la de lugar, tampouco mudar seus valores interativamente.

Tipos de luzes disponíveis dentro da Unreal

Como no nosso caso estamos, a princípio criando um cenário estático, onde o usuário irá somente olhar em volta, sem tocar nada, podemos fazer toda a nossa iluminação com luzes estáticas, e gerar a maior qualidade possível nos aproveitando do bake de luz. Caso se deseje criar uma experiência mais interativa, é possível mudar depois essas luzes para dinâmicas ou móveis.

Começamos então pela luz ambiente, a que entrará pelas janelas - ou seja, a luz vinda da lua. Para criá-la, começamos adicionando uma *blueprint* chamada *BP_sky_sphere* que já vem inclusa com a engine. Basta arrastá-la para a *viewport* que será adicionado um céu interativo, que fica de noite ou de dia, dependendo do que queremos. Para iluminar, no entanto, é preciso adicionar uma luz direcional, que será interligada a essa blueprint. Daremos um leve tom azulado a esta, para que lembre o luar. Após isso, criaremos as luzes internas, aquelas que serão provenientes das chamas das velas. Basta criar uma luz pontual e posicioná-la sobre a vela, e nesse caso daremos um tom alaranjado, característico do fogo. Finalmente, adicionamos uma Skylight, que é responsável por melhorar a qualidade da iluminação interna.

Com isso estamos quase prontos para começar os testes. O processo de baking dos lightmaps é feito por um programa interno chamado Lightmass, mas é preciso fazer alguns ajustes para que ele funcione corretamente. Como nosso cenário é relativamente pequeno, nós queremos que ele concentre os raios de luz, e consequentemente, a qualidade, dentro do ambiente. Para isso criamos um Lightmass Importance Volume e o colocamos em volta dos objetos; além disso, colocamos Lightmass Portals, que servem para guiar raios de luz para ambientes internos, bem em cima das janelas.

Processo de desenvolvimento da iluminação (arquivo pessoal)

Boa parte do processo de iluminação consiste de tentativa e erro. É preciso constantemente reconstruir a luz, enquanto se ajusta paramêtros como intensidade e cor, até que se ache o balanço ideal.

3.5 FINALIZAÇÃO

Um segredo que os designers de efeitos visuais não contam quando estão produzindo um filme, mesmo as grandes produções de Hollywood, é o quão feio um render puro parece e o quanto de “maquiagem” é aplicada na fase de pós-produção para que o resultado final pareça bonito e realista. Por isso, quando estivermos satisfeitos com a iluminação do ambiente, entramos na fase de pós-processamento, que é quando aplicamos efeitos sobre a imagem que a deixarão mais agradável visualmente.

Dentro da Unreal a maior parte desses efeitos fica em um só lugar: no Post Processing Volume. Ele dá a opção de refinar diversos aspectos da cena, como a luminosidade, o balanço de cores, saturação, etc., até mesmo coisas como a intensidade do Ambient Occlusion, das sombras e intensidado do grão. Ajustar esses valores é uma questão mais pessoal, e depende de cada cena. Neste caso, por exemplo, só deixaremos as luzes um pouco mais claras. Só é preciso ter cuidado com o fato de que o Post Processing Volume influencia a cena toda.

Outro efeito que produz um efeito bem interessante em cenas interiores é o Volumetric Fog.
Ele cria luz volumétrica a parte da luz direcional.

Com todos os ajustes feitos, só resta exportar o resultado final para que possa ser visualizado externalmente. Como o trabalho foi todo feito na Unreal Engine, ou seja, um motor gráfico para games, a partir daqui há dois caminhos: gerar um frame parado, como uma foto em 360, para que possa ser visto em qualquer dispositivo com a ajuda de um simples cardboard, ou uma experiência interativa, como um jogo, onde o usuário possa inclusive andar pelo ambiente. Esta segunda opção, no entanto, ainda não é viável para um hardware menos potente, como um smartphone por exemplo, necessitando de um aparelho como um Oculus Rift ou HTC Vive, que são caros e dependem de um computador poderoso para funcionar - fato que em breve irá mudar, obviamente. No entanto, iremos com a primeira opção.

Há diversas maneiras de se tirar uma foto 360 dentro da Unreal, mas a maneira mais fácil é por meio do plugin Ansel, criado pela NVIDIA. Ele já vem embutido no programa, mas é preciso preciso ativá-lo e seguir alguns passos para que esteja pronto para uso.

Com tudo feito, é muito simples utilizá-lo: basta entrar no modo de jogo e apertar o botão de atalho, que ele “para” o jogo e entra em um modo de fotografia, onde é possível ainda ajustar opções como o ângulo da câmera, exposição, etc.

Resultado final, já no formato correto para visualização (arquivo pessoal)

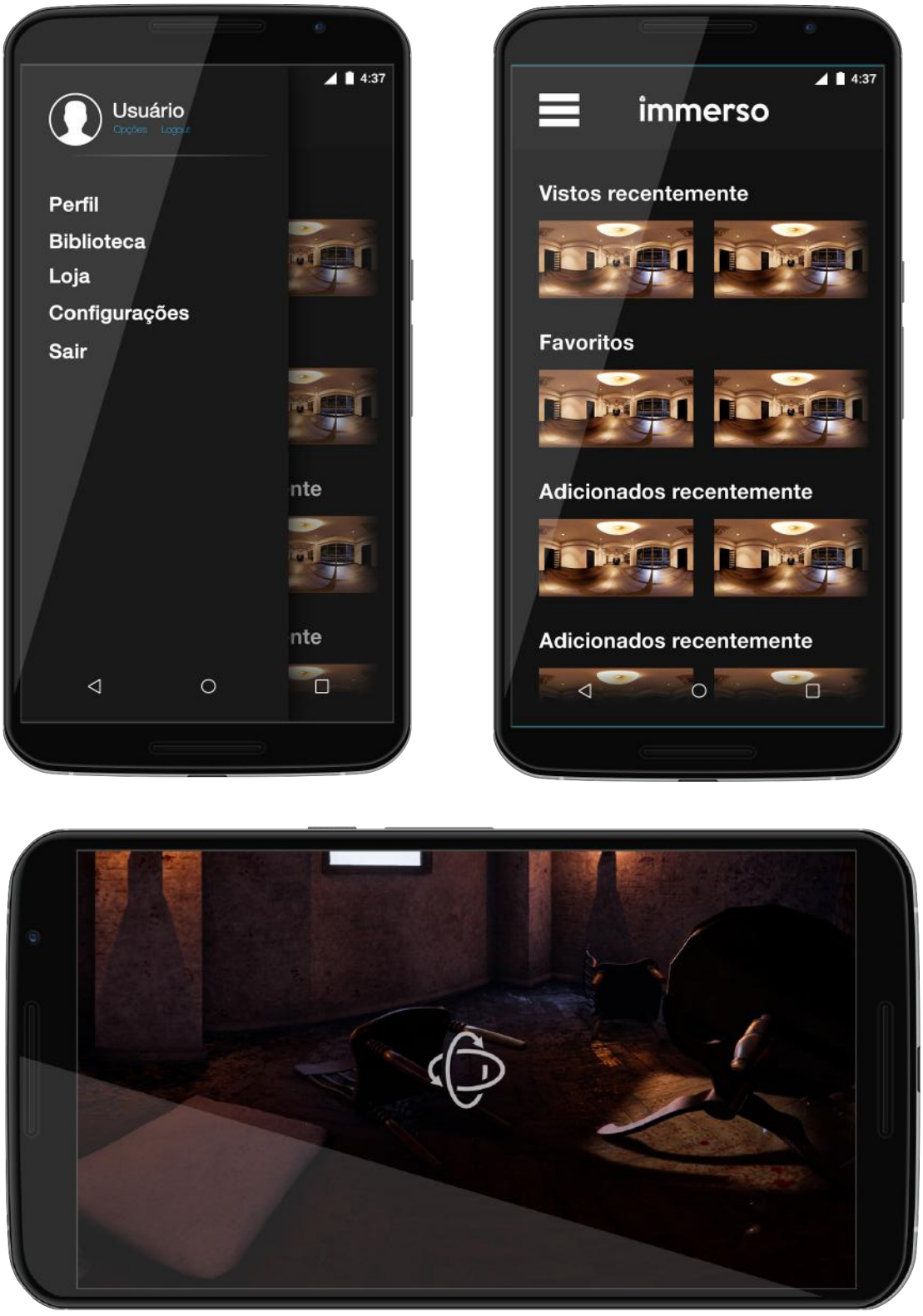
immerso

Com essa imagem gerada, fica a questão: como transformá-la em uma experiência do usuário? Em um mundo tão conectado, como compartilhá-la para que as pessoas possam usufrui-la?

São duas questões a se resolver; a primeira sendo, onde o usuário irá encontrar esses cenários? Irá pegá-los da internet, diretamente com o criador, se for do desejo deste compartilhar seu trabalho? Dessa maneira, muito conteúdo será perdido; a pessoa terá que ativamente buscar esses desenvolvedores, e há muitas variáveis envolvidas até que o usuário tenha acesso ao arquivo. A opção ideal seria que todos os ambientes em realidade virtual que fossem produzidos estivessem em um mesmo lugar, onde seria fácil de se encontrar o que deseja, com recursos para filtragem e busca.

A segunda questão é: feito o download do arquivo, como este será reproduzido? Certamente há dezenas de reprodutores de conteúdo em realidade virtual disponíveis, mas seria realmente a melhor opção sair de um aplicativo, para entrar em outro somente para reproduzir o que você acabou de baixar? É um passo desnecessário a mais, que causará perda de tempo e frustação.

Para resolver essas questões foi pensado no Immerso: um aplicativo de reprodução e compartilhamento de experiências em realidade virtual, tanto para usuários como para desenvolvedores. Para o usuário funcionaria da seguinte maneira: ao abrir o aplicativo, ele encontraria uma loja onde seria possível escolher entre diversas experiências, que poderiam ser filtradas por gênero, autores da obra original, estilos visuais, níveis de interatividade, entre outras opções; feita a escolha, iniciaria-se a reprodução, que poderia ser feita com a ajuda de um visualizador de realidade virtual, como o Google Cardboard, ou diretamente na tela do aparelho celular. Já para os desenvolvedores que desejem criar ambientes como o criado neste projeto, seria possível fazer o upload de suas criações para a plataforma Immerso, para que o mundo pudesse apreciá-las; e inclusive poderia colocar nelas um preço que considerasse justo.



Imagens exemplificando o funcionamento do aplicativo (arquivo pessoal)

4.1 IDENTIDADE VISUAL

“immerso” é um aplicativo que tem como proposta principal a simplicidade; simplicidade de uso, simplicidade no visual, simplicidade na função. A ideia principal é que o usuário possa ir do “abrir o aplicativo” até “reproduzir o conteúdo” da maneira mais fluida e rápida o possível. Por isso ele precisa contar com uma interface minimalista e extremamente fácil de usar. A identidade visual precisa refletir essas características. Ao mesmo tempo, a marca precisa ter personalidade, o que nos leva ao cubo. O cubo é facilmente o primeiro elemento a ser pensando quando se fala em três dimensões, característica que é fundamental a realidade virtual. Pensando nisso, a tipografia do logotipo é a **Keep Calm Medium**, que foi modificada para ter um cubo no lugar do pingo do i. Eis o resultado final:

PALETA DE COR E APLICAÇÃO

A cor do logotipo deverá ser sempre em escala de cinza. Pode ser aplicada sobre cores sólidas, gradientes ou imagens, desde que a leitura seja preservada.

CONCLUSÃO

Como dito no início desta dissertação, o mundo tecnológico está constantemente mudando, e cada vez mais rápido. Como exemplo, durante o processo de criação deste trabalho, a NVIDIA, empresa de tecnologia conhecida por criar as melhores unidades de processamento gráfico (GPUs) do mercado e por estar sempre a frente do desenvolvimento da tecnologia de ponta no mundo computacional, apresentou na Game Developers Conference de, em março de 2018, em São Francisco, um meio para fazer *ray tracing* em tempo real. *Ray tracing*, como o leitor deve se recordar, foi brevemente discutido no início deste texto: é a técnica utilizada pelos renderizadores tradicionais, que simula os raios de luz e sua interação com os objetos, o que permite simular efeitos como sombras e reflexos com altíssima qualidade, mas com a desvantagem de custar um imenso poder computacional. O que a NVIDIA propôs, no entanto, é uma maneira de fazer isso em tempo real; isso virtualmente elimina o principal problema que se tem ao trabalhar com computação gráfica, que são os tempos de espera enquanto um frame é renderizado. Essa nova tecnologia sempre foi considerada o Santo Graal da computação gráfica, e era inimaginável até recentemente. Isso significa também, que alguém que resolver usar este trabalho como inspiração e criar uma experiência em realidade virtual daqui a alguns anos, conseguirá um resultado bem melhor visualmente.

Isso não significa, entretanto, que todos os processos apresentados aqui serão inúteis em breve. Pelo contrário, é grande a possibilidade de que as etapas (modelagem, texturização, iluminação, renderização) continuem seguindo o mesmo padrão; a diferença estará nas limitações que existem atualmente, e que, aos poucos, irão desaparecer, como número de polígonos, resolução das texturas, etc.

Ao final deste trabalho, então, geramos um ambiente completamente virtual, feito do zero, tendo como inspiração uma grande obra da literatura. Se trata de uma nova maneira de se experienciar histórias - as pessoas não irão somente lê-las, mas também estarão dentro delas; irão pisar onde seus personagens favoritos pisaram; irão ver o que eles viram. E além de tudo, foi apresentada uma maneira de levar essa possibilidade ao mundo todo através do aplicativo Immerso.

Porém, mais do que gerar um único resultado, o principal objetivo deste trabalho é, ao menos, proporcionar um mínimo de norteamento àqueles que também querem dar vida as suas histórias preferidas. É meu desejo que os leitores desta dissertação sintam-se encorajados a criar seus próprios universos, e que compartilhem suas criações com todos. Inspiração não falta - é minha esperança que com este trabalho, também não falte os meios de como fazê-lo.

BIBLIOGRAFIA

CARDINAL, David. How Nvidia's RTX Real-Time Ray Tracing Works. 2018. Disponível em < https://www.extremetech.com/extreme/266600-nvidias-rtx-promises-real-time--ray-tracing >. Acesso em 29 de abril de 2018.

CRASH BANDICOOT - LOW POLY (EVOLUTION OF CHARACTERS IN GAMES) - EPISODE 3. 2017. Disponível em < https://www.youtube.com/watch?v=y3Rjg8JGa9w >. Acesso em 13 de março de 2018.

HOW TO CREATE 360 SCREENSHOTS IN UNREAL ENGINE. 2017. Disponível em < https://youtu.be/dbNUPCHxFNg >. Acesso em 14 de abril de 2018.

STEUER, Jonathan. Defining Virtual Reality: Dimensions Determining Telepresence. Journal of Communication. 1992. Disponível em < http://faculty.washington.edu/farkas/TC510-Fall2011/SteuerMediaRichnessTheory.pdf>. Acesso em 02 de maio de 2018.

MARTINS, Simone R.; IMBROISI, Margaret H. Rococó. Disponível em < https://www.historiadasartes.com/nomundo/arte-barroca/rococo/ >. Acesso em 15 de fevereiro de 2018.

RADEMACHER, Paul. Ray Tracing: Graphics for the Masses. Disponível em < https://www.cs.unc.edu/~rademach/xroads-RT/RTarticle.html >. Acesso em 29 de abril de 2018.

UE4 - GLASS MATERIAL HINTS. 2018. Disponível em < https://youtu.be/KQinXCxLOk-Q?list=PLgTiCLtGi8T0t3nM1xCfYRD2HCmF1tOC6 >. Acesso em 19 de março de 2018.

UE4 - HOW TO USE LIGHTMAPS - RULES AND GUIDELINES. 2014. Disponível em < https://youtu.be/gMXye-JhCyQ?list=PLgTiCLtGi8T0t3nM1xCfYRD2HCmF1tOC6 >. Acesso em 14 de janeiro de 2018.