Desenvolvimento de um workflow para a criação de *assets* e gráficos HD-2D para jogos.

Daniel Iop de Menezes, Ricardo Frolich da Silva
Curso de Jogos Digitais
UFN – Universidade Franciscana
Santa Maria– RS

Resumo— Nos primórdios da indústria dos games, o estilo de pixel art era largamente adotado pelos desenvolvedores devido as limitações dos hardwares das épocas passadas. A tecnologia destes hardwares evoluiu, mas o estilo artístico de pixel art não se perdeu. O que antes era uma limitação gráfica passou a ser um estilo artístico apreciado por uma quantidade considerável de jogadores ao redor do mundo. Em títulos recentes, como Octopath Traveler (2019) e Triangle Strategy (2022), uma combinação artística específica é empregada: O uso de assets 3d com texturas e personagens 2. Ambos, somados com algumas escolhas de direção de arte e efeitos especiais formam um estilo artístico conhecido como HD-2D. Este trabalho teve como seu objetivo a busca por um *workflow* que inclua as melhores ferramentas, práticas e passos para se atingir esse estilo artístico dentro do motor gráfico Unreal Engine 5. Para isso, foram feitas diversas pesquisas sobre as melhores ferramentas e técnicas, além de testados os mais diversos *softwares*. Como validação ao *workflow*, foram criados alguns cenários dentro da Unreal Engine 5.

Palavras-chave: HD-2D, Pixel Art, Efeitos de pós processamento, modelagem 3d, Assets.

I. INTRODUÇÃO

Gráficos [2] HD-2D tem por base uma mescla entre elementos 3d e elementos 2d dentro de um motor gráfico 3d. O termo foi cunhado primeiramente em referência ao jogo [1] Octopath Traveler, criado pela Square Enix como uma homenagem à antigos JRPGs, como Chrono Trigger, Live a Live (1994) e Dragon Quest, famosos por sua câmera *top down*, estética em *pixel art* e mecânica de batalhas por turnos.

O uso de *sprites* 2d em planos tridimensionais não é nenhuma novidade. Jogos como Doom Classic se valeram muito dessa técnica como uma forma de agregar estilo artístico ao jogo e manter a otimização dele, tendo em vista as limitações dos antigos hardwares. Além disso, a própria Square Enix, que na época apenas Square, já produziu títulos anteriores que usavam de métodos que mesclam personagens 2d em ambientes 3d, como Xenogears (1998) e Super Mario RPG (1996). Tal prática tornou-se nichada com o tempo, principalmente após o lançamento de *hardwares* como o Playstation 2, capazes de rodar jogos totalmente em 3d e com uma capacidade gráfica superior. Contudo, jogos como o já citado Octopath Traveler (2019), Triangle Strategy (2022) e o remake de Live a Live (2023) tem chamado atenção da parte de consumidores e desenvolvedores que se interessam por tal estética, a qual atingiu o ápice de sua qualidade a ponto de receber um nome próprio: HD-2D. Porém, em matéria de desenvolvimento, existe uma certa falta de suporte no tocante a conteúdos que ensinem adequadamente um caminho para que desenvolvedores criem gráficos HD-2D e uma boa parte do que existe carece de testes e validação adequada para enquanto parte de um *workflow* para desenvolvedores de jogos. Com isso em mente, surgiu a ideia de testar os métodos existentes e as mais diversas ferramentas e técnicas a fim de propor o melhor workflow possível para a criação de *assets* e gráficos HD-2D.

Figure 1. Captura de tela de Octopath Traveler

Este trabalho possui como objetivo a elaboração de um workflow contendo boas práticas e ferramentas gratuitas para a criação de cenários, gráficos e assets HD-2D para jogos digitais. Para validação deste *workflow*, foi feito um estudo de caso que culminou na criação de alguns pequenos cenários e assets.

II. REFERENCIAL TEÓRICO

Neste referencial teórico, será apresentada a pesquisa efetuada para alcançar o estilo artístico HD-2D e os meios empregados para se superar as dificuldades e chegar até o resultado final.

Pixel Art

Pixel art é um tipo de estilo artístico no qual pixels visíveis são utilizados a fim de criar um determinado elemento, seja ele um personagem, objeto ou cenário. O termo foi empregado pela primeira vez em uma publicação de [3] Adele Goldberg e Robert Flegal do centro de pesquisas da Xerox em Palo Alto, Califórnia, EUA, em 1982. Contudo, o conceito em si da arte com *pixels* visíveis é anterior ao termo cunhado, já que jogos com esse estilo artístico, como Space Invaders (1978), foram lançados antes da data da primeira citação do

termo. Existem diferentes tipos, estilos e resoluções de *pixel art*. Alguns jogos, como Octopath Traveller, trabalham com artes mais detalhadas em seu cenário, já desenvolvedores independentes por vezes preferem trabalhar com estilos de *pixel art* mais simples.

Modelagem 3D

Dentro da ciência de computação, modelagem 3d é o processo de desenvolver uma representação através de complexos cálculos matemáticos efetuados por um software de uma superfície tridimensional qualquer. A modelagem 3d teve sua origem em [22] 1963, quando um cientista de computação chamado Ivan Sutherland, Estadunidense, criou um programa chamado Sketchpad como sua tese de doutorado, o qual rodava no computador TX-2 (ou MIT-Lincoln Laboratory TX-2).

HD-2D

O estilo HD-2D é o resultado não só do refino artístico de técnicas que foram empregadas por anos em títulos já citados anteriores ao Octopath Traveler como também da própria evolução tecnológica dos *hardwares*. O principal elemento de um jogo HD-2D é a mescla de *assets* 2d com um ambiente tridimensional. Mas isso por si só não é o bastante para um projeto ser considerado HD-2D. Nesse caso, é preciso definir de forma objetiva quais elementos formam os gráficos HD-2D. De forma simples, os elementos são:

1. O hibridismo entre 2D e 3D, mas aqui lidando com gráficos de alta resolução, nesse caso, 1280x720p (HD) ou maior.
2. Design de personagens e objetos em estilo de pixel art, sendo os personagens e alguns objetos 2d mas o ambiente em si 3d e bem detalhado.
3. Efeitos visuais modernos aplicados, os quais serão explicados em mais detalhes abaixo.
4. Alternância entre os tipos de câmeras fixas e rotativas.

O último elemento, câmera, pode ser tratado com certa liberdade da parte do jogador. Já existem projetos independentes de jogos inspirados por Octopath Traveler que trabalham com câmeras livres, como mostrado pelos canais do YouTube [10] StoneLab Studio e [11] CobraCode. Os próprios efeitos visuais também podem ser alterados de acordo com a vontade do desenvolvedor.

Efeitos de pós processamento

Diversos motores gráficos possuem ferramentas de efeitos de pós-processamento. Eles permitem à artistas e designers aplicarem efeitos gerais para a aparência de uma cena, controlando aspectos como a iluminação, saturação, mapeamento de tom e exposição. Dentro da estética HD-2D, alguns efeitos são recorrentes e característicos. Eles já foram citados previamente, mas é preciso entender o papel que exercem nos gráficos de um jogo. Os efeitos aplicados em Octopath Traveler e comuns à estética HD-2D são: *Ambient Oclusion*, *Lens Flares*, *Motion Blur*, *Vignette*, *Depth of field* e *Bloom*.

Ambient Oclusion é um efeito que faz com que objetos tenham uma exposição realista à luz. Ele permite que objetos projetem sombras de acordo com quanta luz estão recebendo em sua superfície e de qual direção essa luz vem. Abaixo uma imagem que demonstra o efeito na prática.

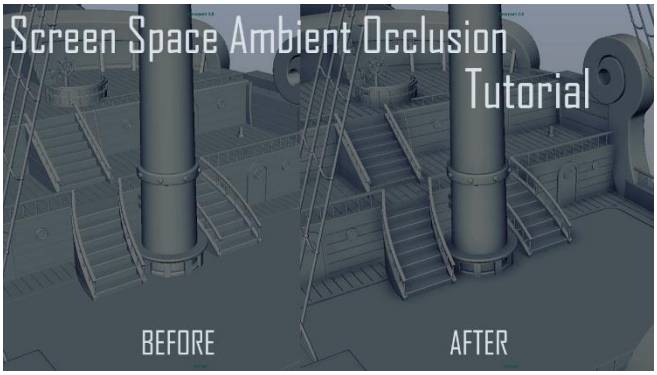

Figura 2. Demonstração do efeito em malha 3d.

Lens Flares é uma espécie de aberração óptica causada através de dispersão de luz em extremidades de lentes. Muito embora esse não seja o método usado dentro de filmes, séries, animações e jogos para aplicar o tal efeito, motores gráficos são capazes de o simular através dos efeitos de pós processamento. O efeito pode variar em forma e intensidade de acordo com a vontade dos desenvolvedores.


Figura 3. Demonstração de lens flares.

Motion blur, ou desfoque de movimento, é um efeito visual famoso entre fotógrafos, sendo capturado pelas lentes ópticas das câmeras fotográficas através de alvos que se movem e uma alta exposição à luz, promovendo assim o efeito de desfoque. Dentro dos jogos, esse efeito é recorrente principalmente entre jogos totalmente 3d, mas é parte da estética HD-2D também.


Figure 4. Demonstração de motion Blur.

Vignette, ou vinheta, é um simples efeito que obscurece as bordas de uma tela a fim de colocar em evidência aquilo que está no centro. Pode variar em intensidade e espaço que ocupa na tela, mas normalmente é usada de forma suave.

Depth of field, ou profundidade de campo, é um fenômeno óptico que descreve até que ponto objetos são vistos com nitidez da parte de um observador. Dentro dos jogos, tal efeito

é simulado pelos motores gráficos a fim de proporcionar mais realismo e promover desfoque de objetos que estão muito longe, consequentemente trazendo para maior foco os que estão mais perto.

Figure 5. Cenário ao fundo se encontra sob desfoque.

Bloom é um efeito existente apenas dentro do campo do pós-processamento, haja vista que ele não é um fenômeno que ocorre de formas naturais, como a profundidade de campo. Este efeito faz com que a luz das bordas iluminadas de objetos se expanda, criando a ilusão de um ambiente muito brilhante e bem iluminado. Abaixo uma demonstração da influência que o efeito exerce sobre uma cena.

Figura 6. Demonstração de bloom

Design Thinking

Design Thinking é um método que tem por base a noção de *design* como uma forma de resolução de problemas. Em suma, é um método de resolver problemas através do estímulo, perspicácia, análise de conhecimentos e proposta de soluções. A ideia de design como um jeito de pensar tem sua origem a partir de 1969, no livro [5] The Science of the Artificial. Mas o conceito de *design thinking* em si só foi cunhado por [6] Rolf Faste, professor na universidade de Stanford. O conceito é atrativo tanto no meio acadêmico quanto no mundo dos negócios por sua capacidade de aplicações variadas e de propor soluções para problemas complexos, como a criação de produtos e serviços. O *design thinking* contempla as seguintes etapas em sua estrutura: Imersão, Análise e Síntese, Ideação e prototipagem.

Imersão consiste de duas sub-etapas, a imersão preliminar e a profunda. Na preliminar, o problema que se deseja solucionar é compreendido, assim como o escopo e limites do projeto. Com isso, é possível avançar para a imersão profunda, na qual se vai até a raiz dos problemas. Nessa fase, podem ser efetuadas entrevistas de emprego, testes de ferramentas, pesquisas de campo etc.

Análise e Síntese é a etapa na qual os dados previamente coletados são organizados de forma coerente e lógica a fim de se entender de forma ainda mais completa o problema.
Na ideação, as primeiras ideias são concebidas a fim de buscar-se as bases para a próxima etapa. Aqui é onde brainstorm e ideias ousadas são bem vindas, assim como possíveis metodologias e *workflows*.

Por fim, na prototipação, validam-se as ideias previamente concebidas e aceitas através de um protótipo, saindo do abstrato e partindo para algo concreto e funcional. Embora colocado como a última etapa, a prototipação pode permear todo um projeto, ocorrendo de forma simultânea com as demais etapas previamente citadas.

Ferramentas de desenvolvimento

Para a criação dos *assets* e cenários apresentados, foram empregados diversos *softwares* e técnicas. Nessa sessão estes serão citados.

Softwares de modelagem 3d

São programas que simulam um objeto tridimensional através de software especializado, permitindo ao usuário manipular as superfícies deste objeto. Três deles foram testados ao longo do desenvolvimento, sendo eles: [8] Blender, [13] Magicavoxel e [14] Blockbench. O último se mostrou o mais adequado de todos não só pela garantia do efeito *pixel perfect* mas também por ser feito para a modelagem 3d com *pixel art*, sendo extremamente popular da parte dos criadores de *mods*, ou seja, conteúdos modificados para jogos de videogame, de Minecraft. Abaixo uma demonstração de um objeto cujas texturas não estão em *pixel perfection*.

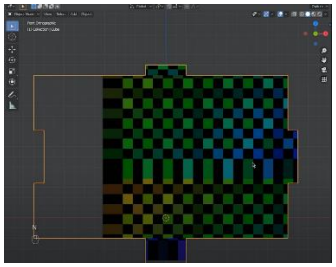

Figure 7. Objeto no blender com discrepância entre os pixels

Experimentos com os outros dois programas citados resultaram em *assets* inadequados. No caso do Blender, a falta de uma *feature* nativa de *pixel perfection* se mostrou um grande devorador de tempo, exigindo ajuste de cada textura em cada face dos objetos 3d. Blender dispõe de alguns plugins que auxiliam a modelagem 3d com *pixel art*, como o [7] Sprytile, [8] Texel Density Checker e [10] Pribambase. Contudo, estes *plugins* funcionam apenas em versões antigas do Blender (que está na sua versão 4.0 no momento da escrita deste texto) e, apesar da qualidade individual de cada um deles, o usuário ainda precisa adaptar uma ferramenta feita para modelagem 3d convencional e renderizações de alta resolução para uma ferramenta de modelagem 3d com texturas em *pixel art*.

Já no caso do MagicaVoxel, sua forma de fazer exportação para *engines* é altamente destrutiva. Através deste *sotfware*, o usuário pode construir cenas 3d com *voxels*, que são *pixels* 3d, ou seja, pequenos quadrados. Este programa trabalha com a extensão de arquivo .vox, que é uma extensão própria. Ao exportar, é preciso converter tal extensão para os formatos .fbx ou .obj, que são os formatos mais comuns na modelagem

3d. Ao exportar, a malha dos modelos criados é desnecessariamente repleta de polígonos. Porém, mesmo que se corrija tal problema, há ainda o problema das texturas embebidas no objeto. Uma vez extraídas, elas se mostram distorcidas e em resolução que não é própria à de uma *pixel art*. Portanto, não são adequadas para uso em jogos. Abaixo uma imagem de uma textura extraída através do Blender de um modelo criado com MagicaVoxel. Ambos os problemas podem ser vistos na imagem abaixo.

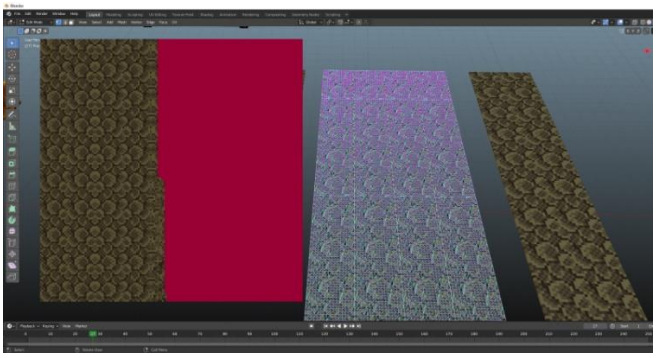

Figure 8. Modelo e textura inadequados

Portanto, como ferramenta de modelagem 3d para este projeto, seria preciso escolher tanto uma ferramenta que permitisse lidar de forma dinâmica com a necessidade de *pixel perfection* como que permita criar texturas de forma não destrutiva. Em ambos os casos, o *software* de modelagem 3d Blockbench atende às expectativas, permitindo uma modelagem que leva em conta o tamanho de um objeto em *pixels* e uma texturização dinâmica, podendo ser feita diretamente através de ferramentas de pintura dentro do software ou de texturas prontas.

Criação de pixel art/texturas

Existem diferentes formas de se obter uma arte em *pixel art*. Diversas bibliotecas, algumas pagas outras gratuitas, fornecem *pixel arts*. A mais destacável delas é o site Itch.io ou então OpenGameArt.org. É possível ainda criar uma textura em *pixel art* do zero através de algum software de desenho feito para tal ou que ao menos seja capaz de desenhar *pixel arts*. Existem muitos deles disponíveis no mercado. Um dos mais famosos e aquele que foi empregado neste projeto foi o [23] Aseprite. Embora ele seja uma ferramenta paga, a escolha de *software* de ilustração gráfica em *pixel art* é absolutamente facultativa, o artista responsável pelas texturas pode sentir-se à vontade para utilizar qualquer programa capaz de fazer *pixel art* com o qual ele possua familiaridade ou preferência.

Como uma escolha de *design* artístico, é possível levar em conta o uso de mapas de texturas. Existem muitos tipos distintos de mapas que podem ser aplicados dentro de um modelo 3d e cada um deles representa uma forma de leitura que o motor gráfico fará sobre algum aspecto, como a profundidade ou rugosidade. Alguns dos materiais (ou *materials*, em inglês) existentes são: *Difuse*, *Albedo*, *Ambiente Oclusion*, *Normal map*, *Displacement*, *Specular* e *Roughness*. Abaixo uma ilustração desses mapas.

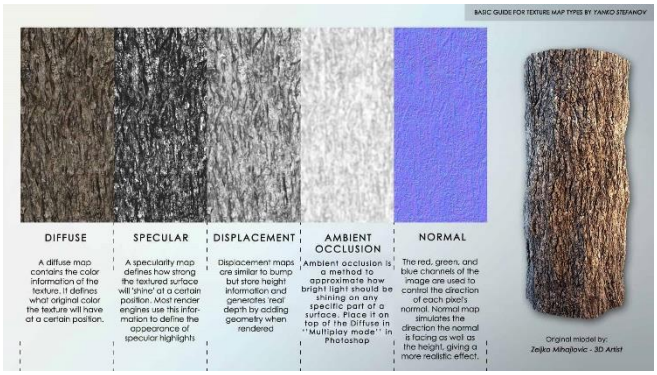

Figura 9. Exemplo de mapas e suas funções

Para este projeto, apenas *Difuse*, *Normal* e *Specular map* foram aplicados nos modelos por questões de escolha artística. O programa utilizado neste caso foi o [16] Laigter por ser a única alternativa gratuita capaz de criar mapas para texturas em *pixel art* com relativa qualidade.

Motores gráficos / Engine.

São *softwares* aplicados no desenvolvimento de jogos, mas alguns também possuem aplicações gráficas diversas, como a [21] Unreal Engine, que é também aplicada para peças de cinema e arquitetura. Estes programas vêm com os mais diversos tipos de ferramentas, como simulação de físicas, criação de animações, ferramentas de programação etc. Unreal Engine foi o motor usado neste processo, mas antes dela a [17] Unity 3d fora testada. É possível, alternativamente, adotar a Unity 3d como motor gráfico para um projeto HD-2D.

III. TRABALHOS CORRELATOS

Desenvolvimento de um workflow para a criação de cenários Foto Realistas para Jogos.

Em seu trabalho de TCC de 2021 pela Universidade Franciscana de Santa Maria, RS, [15] Victor efetuou um estudo de natureza semelhante a este trabalho, porém, aplicado para um estilo artístico totalmente distinto: o fotorealismo. Em seu TCC, Victor realizou pesquisas históricas e técnicas acerca do fotorealismo. Muito embora os trabalhos sejam diferentes na matéria que eles abordam, a metodologia por este trabalho empregada serviu de base para a produção do vigente projeto.

Bioblox 2.5D — Developing an Educational Game Based on Protein Docking.

Neste trabalho feito por estudantes da University of London e Imperial College of London, desenvolveu-se uma aplicação na forma de um jogo educativo cujo intento é, através de um [20] jogo de *puzzles*, ensinar alunos de biologia sobre docking, um processo biológico efetuado por proteínas. Este projeto empregou Autodesk Maya como seu modelador 3D, Unity 3D como seu motor gráfico e Photoshop como programa para desenhos. As mesmas ferramentas tentaram ser aplicadas inicialmente neste projeto, as quais se mostraram ineficiente para o contexto deste projeto. Ainda assim, os primeiros passos para a devida compreensão do artigo redigido foram dados através desta primeira iteração proporcionada pela apreciação deste trabalho correlato.

IV. IDENTIFICAÇÃO DO PROBLEMA E PROPOSTA DA SOLUÇÃO

Como já mencionado antes, a ausência de um conteúdo contundente no tocante ao passo a passo para a criação de arte no estilo HD-2D e que permita ao desenvolvedor a segurança de estar apoiado por um *workflow* sólido é o maior problema para a criação de jogos no estilo HD-2D, assim como a dificuldade de se encontrar e empregar as ferramentas certas para desenvolver esse tipo de jogo da forma mais eficiente possível. Portanto, este trabalho se propõe, através das experimentações realizadas, a criar um workflow detalhado desde a concepção de ideias até o momento em que os *assets* são colocados para funcionar dentro de um motor gráfico.

V. METODOLOGIA

Este trabalho tem por seu objetivo a criação de um *workflow* para a produção de assets e gráficos HD-2D, podendo este *workflow* vir a sofrer alterações uma vez adotado por desenvolvedores, de acordo com as demandas de seus projetos. Como guia para a busca pelas melhores ferramentas e métodos a serem contemplados pelo workflow proposto, foi aplicada a metodologia de [18] Chandler, a qual permite o processo de iterações.

Inicialmente foram estudadas as bases que formam a estética HD-2D, depois compreendidas as técnicas de criação de *pixel art* e, *a posteriori*, analisados os tipos de modelagem e estilos de *pixel art* empregados pelos mais diversos jogos HD-2D no mercado. Então foram testadas diversas ferramentas a fim de encontrar as mais eficientes, priorizando ferramentas gratuitas nesta busca. Uma vez validadas as ferramentas, como forma de pautar a busca pelo *workflow*, foram estudados dois outros *workflows*. Sendo eles o *Design Thinking* e o workflow segundo Wiktor Öhman, apresentado no trabalho "Desenvolvimento de um Workflow para a Criação de Cenários Foto Realistas para Jogos", ambos citados no referencial teórico. Como forma de validação à este *workflow* final, foram criados diversos *assets* e montados ao todo quatro pequenos cenários.

Workflow

Um *workflow* ou *pipeline* de produção nada mais é que um fluxo de trabalho no qual ocorre o desenvolvimento de um projeto do conceito mais básico até sua finalização. Possuir um *workflow* como forma de solucionar um determinados problemas é uma abordagem largamente adotada dentro de empresas quando o assunto é a resolução de problemas complexos. Dentro do campo de jogos, ter um *workflow* significa saber que passos tomar até chegar no produto final. No caso específico deste projeto, significou também conhecer as ferramentas certas para o trabalho a fim de torná-lo mais eficiente em matéria de tempo. Por tomar o *Design Thinking* como base, este workflow pode ser executado de forma não linear, permitindo que etapas ocorram simultaneamente ou sejam revistas em caso de resultados abaixo das expectativas. Este *workflow* contém oito etapas, que podem ser utilizados qualquer disposição de design que se adote com relação ao estilo HD-2D. Suas primeiras etapas são mais abrangentes e conforme se avança o *workflow* torna-se mais objetivo.

Primeiro passo: Análise

A análise é a etapa na qual se fará um levantamento teórico acerca dos problemas que requerem solução, bem como a limitação do escopo do projeto a ser executado. Nessa etapa, contempla-se as necessidades primárias do projeto, tais como modelagem 3d, texturização, *assets* e até mesmo decisões de *design* básicas, como quanto ao tipo de cenário que se deseja construir, se maior ou menor, aberto ou fechado, linear ou não linear, gráficos caprichados ou simples etc.

Segundo passo: Requisitos

Conhecendo os problemas, é preciso buscar as ferramentas certas para o trabalho. É sobre isso que essa etapa se trata. Aqui, é feito os levantamentos acerca de ferramentas necessárias para o projeto. Isso significa a escolha de motor gráfico, *software* de modelagem 3d, ilustração gráfica, programação e qualquer outro tipo de programa que possa ser necessário na produção de um projeto.

Primeiro passo: Ideação

A ideação é a etapa onde as coisas começam a se tornar mais objetivas e específicas. Uma vez que a decisão de um determinado estilo de produção foi adotada, buscam-se as referências, artes conceituais e temática para dar forma palpável ao projeto em questão. Portanto, a ideação torna seu escopo ainda mais limitado, lhe permitindo focar naquilo que é de fato o objetivo do projeto. Uma forma simples de conseguir referências e conceitos caso não se tenha acesso à um artista conceitual é usar de sites como [19] ArtStation, os quais podem ser fontes de referências e ideias.

Terceiro passo: Blocagem

Aqui é onde o 3d começa a entrar em cena, uma vez que é parte fundamental da estética HD-2D. Uma blocagem nada mais é do que fazer um esboço com objetos 3d. Ela é fundamental para se definir a disposição espacial, escala, arranjos básicos e ter a sensação inicial de como os objetos na cena irão se comportar em conjunto. Fazer uma blocagem é sobre peças simples e rápidas de serem feitas. Portanto, não se sinta 100% preso a seu conceito. Se uma determinada distribuição de elementos não ficou boa, sinta-se livre para tentar outra.

Quarto passo: Modelagem e textura

Com a blocagem concluída, é a vez dos modelos tomarem sua forma final e receberam suas texturas e mapas de texturas. Uma prática interessante aqui é buscar reusar texturas de uma forma orgânica, assim fazendo com que a repetição não soe como algo monótono e enjoativo para sua cena. Pequenas modificações em uma determinada textura que se repete podem ser a salvação para a boa aparência de seu cenário. Outra boa prática é fazer versões básicas das texturas e só depois que essas texturas se mostrarem ao nível das expectativas do projeto as concluir e adicionar ao cenário.

Quinto passo: Visualização

Nesta etapa, você irá produzir a câmera de seu jogo ou cena dentro do motor gráfico escolhido. Jogos no estilo HD-2D costumam usar câmeras isométricas ou fixas, mas é possível o uso de uma câmera diferente, como a câmera em terceira pessoa padrão. De qualquer forma, levar a câmera em consideração na hora de distribuir os objetos na cena é fundamental. Caso contrário, o projeto provocará uma clara estranheza. Em um cenário isométrico, por exemplo, espera-se ser capaz de enxergar tudo de cima. Fazer modelos muito

altos que não se encaixem com esse tipo de câmera simplesmente não é bom para este projeto ficcional. Isso deve ser levado em conta principalmente na ideação e no conceito

Sexto passo: Efeitos

Aqui se trata dos efeitos de pós processamento de um motor gráfico, que são efeitos especiais aplicados dentro de uma cena, os quais permitem modificar a luz, sombra e a forma como eles interagem no cenário. Existem muitos efeitos, mas apenas alguns são comuns à estética HD-2D.

Sétimo passo: Otimização

A natureza da modelagem e texturas empregadas neste estilo artístico são muito mais leves se comparados com modelos e texturas normais. Contudo, descuidos principalmente na área dos efeitos de pós-processamento podem levar a um projeto mau otimizado. É recomendado sempre manter olhos abertos sobre os FPS (*frames* por segundo) de seu projeto e seu peso computacional. Algumas dicas aqui são: Desligar as sombras de objetos nos quais elas não fazem diferença e deixar de renderizar objetos que não aparecem na tela do jogo, algo que já vem por padrão no motor gráfico Unreal Engine. Em casos extremos, como por exemplo de um jogo com muita extensão de terreno, como um mundo aberto, utilizar- se do processo de *baking* 3d é uma boa pedida para aliviar o peso computacional do projeto, algo que não foi feito neste caso concreto por não haver necessidade de otimização, já que o projeto não será aplicado, ao menos não com a distribuição espacial atual, em um jogo concreto.

VI. ESTUDO DE CASO

Para validar o *workflow* proposto neste trabalho como um *workflow* viável para a produção de assets HD-2D usáveis na produção de jogos, foram produzidos alguns assets 3d e colocados dentro da Unreal Engine 5 com os respectivos efeitos de pós processamento já citados. Unreal Engine 5 foi a escolha final para esse estudo de caso ao invés de Unity3d devido a facilidade de se aplicar efeitos de pós processamento, fazendo do motor mais amigável para artistas. Contudo, em havendo conhecimentos de aplicação de efeitos de pós processamento na Unity3d, este motor gráfico torna-se perfeitamente viável para a produção de jogos HD-2D. Abaixo serão explanadas as abordagens e soluções encontradas no tocante a cada uma das etapas do projeto dentro do *workflow* proposto.

Análise

De forma breve e resumida, foi decidido que seriam feitos quatro pequenos cenários e que estes seriam feitos de forma totalmente tridimensional, sem apelar para o uso de imagens 2d como objetos no cenário. Por se tratar do estilo HD-2D, pixel art certamente estaria inclusa como o modelo de arte de cada um dos objetos. Além disso, logo de cara, fora decidido que as cenas finais, as quais não se tem intenção de serem empregadas em um jogo de fato, precisariam estar munidas de efeitos de pós-processamento tais quais os empregados em Octopath Traveler.

Ideação

Como conceito básico dos assets criados, foi decidido que haveria dois conjuntos de assets: Um para um cenário exterior com árvores, vegetação e casas e outro para interiores, representando assim uma habitação e dois estabelecimentos, respectivamente, ambos partindo de uma estética clássica de rpgs de fantasia medieval. Foi também decidido previamente que a densidade de pixels adotada dentro do Blockbench seria de 16x16.

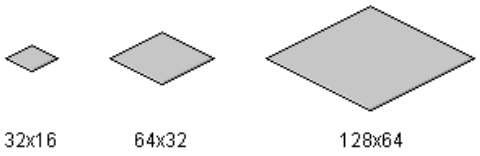

Figure 10. Quanto maior a resolução, maior o detalhamento.

Conceitos e referências

Foram escolhidas capturas de tela do jogo Octopath Traveller para servirem de inspiração para os *assets* e a forma como eles viriam a ser distribuídos dentro da Unreal Engine. Duas para retratar estabelecimentos internos, uma para o interior de um quarto e outra para representar um cenário externo, contendo casas, árvores, grama etc.

Blocagem

Devido a escolha da Unreal Engine e sua falta de acesso a ferramentas de *Tilemap* 3d, ao contrário da Unity, a blocagem do cenário foi feita no software de modelagem 3d. Apesar dessa escolha, é possível sim fazer blocagens dentro da Unreal Engine através do sistema de *Geometry*, que permite criar objetos 3d editáveis em tempo real dentro do motor gráfico. Na versão 5 da Unreal Engine, que é utilizada nesse projeto, é ainda possível valer-se de ferramentas de modelagem e texturização mais avançadas nativas da própria ferramenta para blocagem

Modelagem

Devido à inexistência de vastas bibliotecas gratuitas de *assets* HD-2D, ao contrário do que se observa com *assets* em outros estilos artísticos, foi preciso produzir tudo manualmente. Sendo assim, através dos programas já citados, os modelos 3d previamente feitos como simplesmente blocagens receberam sua forma final e suas devidas texturas. Para a criação dos mapas de textura, *normal* e *specular maps*, respectivamente, foi usado programa gratuito Laigter. No que diz respeito ao ambiente da cena, a Square Enix desenvolveu para si uma ferramenta na Unreal Engine 4 que proporcionava a possibilidade de se moldar terreno com base em blocos e quinas, assim formando o ambiente de Octopath Traveller. Infelizmente tal ferramenta nunca foi disponibilizada para público e, até o presente momento, não há qualquer ferramenta dentro da própria Unreal Engine que proporcione algo semelhante. Portanto, os *assets* que constituem as cenas internas foram construídos modularmente e adicionados de forma manual na cena.

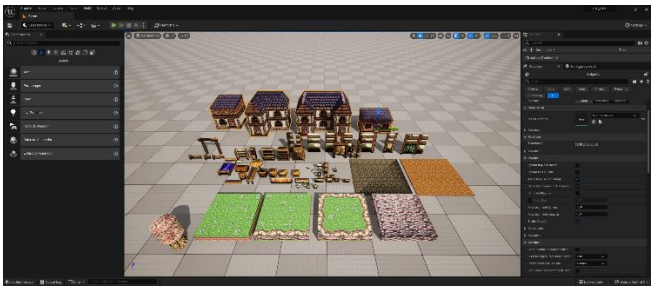

Figure 11. Assets em suas formas finais.

Visualização

Para o caso concreto do cenário produzido através deste *workflow*, ele fora feito pensando em uma câmera lateral presa, mas as opções são muitas, conforme mostrado através dos jogos citados como exemplo ao longo deste texto. Contudo, por não haver a intenção de aplicação da cena em um cenário real de jogo, recursos programáticos, como movimento de câmera e personagem para que essa câmera siga não foram produzidos. Ainda assim, o posicionamento da câmera foi levado em conta o tempo todo desde a análise até produção a finalização do projeto.

Efeitos

Para os critérios do presente projeto, seguiram-se alguns dos mesmos efeitos aplicados no jogo Octopath Traveller, sendo eles: *Ambient Oclusion*, *Vinheta*, *Depth of Field*, *Bloom*, *Manual Exposure*. Todos estes efeitos são explicados em detalhes dentro da documentação da Unreal Engine e são alcançáveis na Unity 3d. Todos estes efeitos foram descritos anteriormente e seus detalhes podem ser vistos na [21] Documentação oficial da Unreal Engine 5.

Figure 12. Um dos cenários produzidos através do workflow

Otimização

Apesar das sugestões feitas anteriormente, práticas de otimização não foram necessárias neste projeto devido a sua natureza e ao fato de que ele não fora feito pensando em rodar um jogo específico.

VII. CONSIDERAÇÕES FINAIS

A estética HD-2D tem retornado aos jogos e sido frequente principalmente entre desenvolvedores independentes. A tendência é que cada vez mais essa estética se torne recorrente em jogos de forma que novas ferramentas dedicadas para a criação desse tipo de jogo venham a surgir. Infelizmente não existem tantos conteúdos online dedicados a propor um *workflow* detalhado de criação de jogos HD-2D, se restringindo em sua maioria a *showcases*/demonstrações de criações feitas por desenvolvedores proprietários de canais no Youtube ou perfis de sites como Artstation. Para auxiliar na escolha de ferramentas e estéticas, foi proposto este workflow. Ele, por sua vez, se mostrou eficiente na hora da produção dos *assets* e sua aplicação dentro da Unreal Engine. Com algumas adaptações, ele pode ser aplicado aos mais diversos estilos de jogos HD-2D e até mesmo para jogos totalmente 3D que desejem utilizar personagens 3d com texturas em pixel art. O resultado pode ser encontrado em: https://drive.google.com/drive/folders/1ZGDpJBiFoIHxhaXvHULcuHQAwVemyPDc?usp=drive_link

Referências

[1] Review: Octopath Traveler. Nostalgia encontra gráficos modernos em um JRPG encantador e de batalhas brilhantes. 2018. Em: https://www.theenemy.com.br/nintendo/criticas/review-octopath-traveler

[2] A arte e a ciência por trás da tecnologia HD-2D — Parte 1: de Super Mario RPG a Octopath Traveler. Em (https://nintendoboy.com.br/2022/07/a-arte-e-a-ciencia-por-tras-da-tecnologia-hd-2d-parte-1-de-super-mario-rpg-a-octopath-traveler/)

[3] A mais poderosa ferramenta 3D em tempo real – Unreal Engine. Em (https://www.unrealengine.com/pt-BR/)

[4] Pixel Art / Arteprog e Programação. Em: (https://arteprog.space/programacao-criativa/conteudo/pixel-arte.html)

[5] SIMON, Herbert (1969). The Sciences of the Artificial. Cambridge: MIT Press.

[6] Patnaik, Dev, "Forget Design Thinking and Try Hybrid Thinking", Fast Company, August 25, 2009. “

[7] Sprytile by Jeiel Aranal. Em (https://jeiel.itch.io/sprytile)

[8] Blender Addon: Texel Density Checker 2023.1. Em: (https://mrven.gumroad.com/l/CEIOR)

[9] Pribambase by lampy. Em: (https://lampysprites.itch.io/pribambase)

[10] StoneLab Studio. Em: https://www.youtube.com/channel/UC5hOu2JFOz5U4Z1g5mdh_HQ

[11] CobraCode. Em: (https://www.youtube.com/@CobraCode)

[12] Blender.org – Home of the Blender Project – Free and Open 3d Modeling Software. Em (https://www.blender.org/)

[13] MagicaVoxel. Em (https://ephtracy.github.io/)

[14] Blockbench – A low-poly 3d Model editor. Em: (https://www.blockbench.net/)

[15] Victor Cesário Coletto. Desenvolvimento de um Workflow para a Criação de Cenários Foto Realistas para Jogos. Em: https://tfgonline.lapinf.ufn.edu.br/media/midias/Victor_Coletto_TFGII_-_Vers%C3%A3o_Final.pdf

[16] Laigter by Azagaia. Em: (https://azagaya.itch.io/laigter)

[17] Plataforma de desenvolvimento em tempo real do Unity. Em: (https://unity.com/pt)

[18] Heather M Chandler. Manual de produção de jogos digitais. Bookman Editora, 2009.

[19] Ballistiq Digital. “ArtStation”. Em: (https://www.artstation.com/)

[20] Bioblox 2.5D — Developing an Educational Game Based on Protein Docking. Em: (https://arxiv.org/pdf/2204.12425.pdf)

[21] Unreal Engine 5.3 Documentation. Em: (https://docs.unrealengine.com/5.3/en-US/)

[22] Sketchpad – primeira interface gráfica avançada de usuário de 1963. Em: (https://museucapixaba.com.br/hoje/sketchpad-primeira-interface-grafica-avancada-de-usuario-de-1963/)

[23] Aseprite. Em: (https://www.aseprite.org/)