LIL'WITCH

A Design Case Study of Animation in Games

Universidade de Coimbra
Mestrado em Design e Multimédia
Faculdade de Ciências e Tecnologia
Gustavo Campos de São Pedro Barroso
Setembro de 2016

Lil'Witch
A Design Case Study of Animation in Games

Universidade de Coimbra
Mestrado em Design e Multimédia
Faculdade de Ciências e Tecnologia
Gustavo Campos de São Pedro Barroso
Setembro de 2016

Orientação
Licinio Roque
Antonio Manuel Sucena Silveira Gomes

Júri
Bruna Raquel Santos Sousa
Juri Vogal: Luís Alegre

Índice

1 Introdução

Em virtude da evolução da tecnologia dos computadores e da criação de novas técnicas, o custo e tempo de produção de animação 3D tem diminuído e os resultados melhoraram tornando-se mais realistas. A animação adquiriu maior complexidade exigindo múltiplas e mais profundas competências que exigem a quem quer abordar esta nova tecnologia, um estudo/formação aprofundado e permanente e recursos dispendiosos. Já a Animação 2D apesar de ter uma coleção de técnicas refinadas, é facilmente acessível a animadores principiantes. Recentemente, têm sido publicados, com sucesso, produtos cinematográficos ou jogos, que combinam Animação 3D com Animação 2D. Esta combinação abre novos horizontes no campo de Animação, temática que esta dissertação pretende explorar, estudar e aplicar num jogo.

1.1 Enquadramento

Esta dissertação enquadra-se na temática de design de jogos, sendo uma área constituída por diversos campos de estudo, de que são exemplo: Áudio, Iluminação, Modelação... Inclui, igualmente, um vasto campo de animação, cuja produção pode ser adaptável às necessidades da equipa de produção, podendo por exemplo ser simples ou complexa, rápida ou morosa, exigindo assim diversas cargas de trabalho. Um jogo constitui uma plataforma ideal para explorar a combinação da Animação 2D com a 3D, na medida em que os jogadores têm mais facilidade em imergir com o jogo sendo, assim, mais sensíveis à animação das personagens que controlam.

1.2 Motivação

Numa época em que as pessoas têm acesso aos recursos básicos para criar animações e jogos, a coleção de técnicas que combinam 2D com 3D, pode constituir uma ajuda ou fonte de inspiração para que criem o seu próprio produto. Enquanto o nível do jogo é um exemplo do resultado das técnicas utilizadas, o seu processo demonstra o potencial dessas mesmas técnicas. Através da consulta da coleção de técnicas e ao jogar Lil'Witch os jogadores podem ficar mais conscientes dos diferentes modos de implementação das técnicas noutros jogos, incentivando-os a explorar o potencial da coleção de técnicas permitindo-lhes criar jogos e animações mais interessantes.
A pesquisa de técnicas 3D e 2D, constitui simultaneamente, uma oportunidade para um aluno de Design e Multimédia estudar e experimentar Multimédia 3D e animação com um grau de profundidade não disponível no curso.

1.3 Objectivo

O objetivo desta dissertação é pesquisar, colecionar e processar informação de animação, do processo de animação 2D e 3D e das técnicas que combinam 2D com 3D, incluindo os produtos em que estas são aplicadas. Dessas técnica, selecionei aquelas, que me permitiram a construção de um nível curto do jogo Lil'Witch. Os critérios subjacentes a esta seleção, foram se tinha a tecnologia, conhecimento e a experiência necessária para as aplicar no jogo sem prejudicar a sua eficiência. Pretendia deste modo, entreter e cativar os jogadores, e, simultaneamente dar a conhecer as técnicas que combinam 3D com 2D e o seu potencial.

1.4 Estrutura do Documento

A Estrutura desta dissertação é dividida nos seguintes cápitulos:

A Introdução que apresenta uma pequena explicação da Tese, incluindo os seus os objectivo, a motivação e como vai ser posto em prática.
O Estado da Arte, que contém a informação pesquisada ao longo da dissertação relevante à dissertação, tal como conceitos-chaves, tecnologias e técnicas.
Abordagem metodologia, explicita o processo para atingir os objetivos propostos pela Dissertação, incluindo a calendarização e desafios do desenvolvimento da prova de conceito, o jogo.
A Proposta de Trabalho indica o método de implementação, o conceito, design e as técnicas que se pretende pôr em prática para criar o protótipo do jogo.
A Implementação do jogo contem o processo de criação do jogo, tal como o design das animações e personagens, as técnicas e tecnologias implementadas do protótipo do Jogo.
A Conclusão que encerra esta dissertação com a reflação do seu resultado, as dificuldades encontradas e as perspectivas futuras.

2. Estado da Arte

Neste capítulo, serão abordados vários temas que servirão como base de compreensão das técnicas de animação usadas nas áreas 2D, 3D e jogos. Este capítulo está dividido em três subcapítulos:
- O primeiro aborda os princípios básicos de animação e começa com o procedimento básico da criação de animação. São definidos os 12 Princípios Básicos de Animação e mencionadas algumas ferramentas de suporte do planeamento da animação.
- O segundo é dedicado ao procedimento de construção do projeto 3D - pipeline 3, especificamente da análise dos vários passos da fase de Produção, tais como Modelação, Textura, *Rigging*, Animação, *VFX*, Iluminação e Renderização. Nestes passos também se incluíram algumas das características especificas dos jogos
- O terceiro é composto pela recolha e a análise de técnicas de animação, incluindo as suas vantagens, desvantagens e exemplos de projetos que exibem o resultado destas técnicas.

2.1 Animação - princípios básicos

"For some presumptous reason, man feels the need to create something of his own that appears to be living, that has an inner strenght, a vitality, a separate identity - something that speaks out with authority-- a creation that gives the ilusion of live."
Thomas F. & Johnston O. (1981) *The illusion of life*. Italy, Walt Disney Productions

Animação é dar vida às personagens e objetos do ecrã. Tecnicamente é uma sequência de imagens, designadas por *frames*, *sprites*, que ilustram uma personagem em diferentes poses. Esta sequência engana a perceção do público levando-o a acreditar que a personagem está em movimento. Tipicamente um vídeo animado tem 24 *frames* por segundo. Como tal um vídeo de 90 minutos teria 129,600 *frames*, logo são 129,600 imagens que têm que ser criadas e analisadas para se certificar que estão consistentes. Portanto, animação é um processo moroso e que exige muito trabalho. Na procura de manter a consistência da ilustração da personagem ao longo dos *frame*, estabeleceu-se um conjunto de princípios para guiar os animadores, que são os 12 Princípios Básicos de Animação, analisados e refinados ao longo do tempo pelos estúdios da Walt Disney. A saber:

Squash and Stretch - É um dos princípios mais importantes, por introduzir certas deformações na personagem ao longo da animação que ajuda a exibir peso e flexibilidade no seu movimento. Um dos exercícios básicos para novos animadores é animar os saltos numa bola, porque é considerado perfeito e simples para testar este principio. Quando a bola cai no chão, por breves momentos fica espalmada no chão, ou seja *squashed*, assim que a bola está prestes a saltar, a sua forma começa a esticar, ou seja *stretch*.

Slow In and Slow Out - Este principio indica que nenhum objeto para imediatamente, ou seja, o animador tem que ter em conta a aceleração e desaceleração em movimentos, que por norma, se exagera em animação. Este principio também pode ser facilmente observado no teste básico dos saltos numa bola. Por exemplo, quando a bola cai, a distância da bola entre as *frames* não é muito grande, logo o seu movimento ainda é lento, mas quando maior for a distância da bola entre *frames* maior é sua velocidade (aceleração).

Anticipation - Indica que no início da animação de uma personagem, a audiência tem que conseguir perceber e antecipar o seu movimento. A criação da antecipação, é possível ao preceder qualquer ação

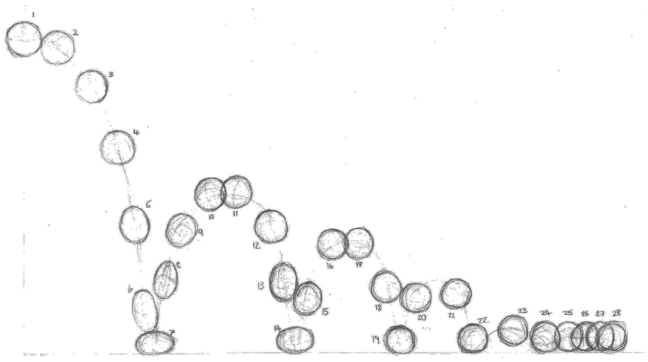


Figura 1- teste básico de animação

da personagem com a introdução de um conjunto de movimentos ou poses no início do seu movimento. Por exemplo, num herói que está prestes a saltar, a ação pode ser antecipada se o herói primeiro se agachar e pôr a sua mão no chão e se estiver a olhar para cima.

Staging - É um principio genérico, presente em diversas áreas, tal como o antigo teatro, televisão... Indica que a apresentação de uma ideia por meios visuais deve ser completamente e inequivocamente clara. De que são exemplo: o artista deve ter cuidado na forma como organiza e compõe o cenário, certificando-se que não existe nenhum elemento que possa roubar a atenção do público em relação às ações da personagem. A câmara deve focar-se na conversa entre personagens e não em elementos do cenário irrelevantes para a narrativa. Em momentos cruciais da narrativa, a pose das personagens deve corresponder à sua expressão, atitude, estado emocional. O sucesso deste exemplo pode ser verificado se a silhueta da pose expressa claramente a emoção da personagem.

Timing- É calcular quanto tempo é necessário para fazer uma ação e a representar em *frames*. Por exemplo, supondo que uma pessoa demora 1,5 segundos para sentar, numa animação de 24 *frames* por segundo, teríamos 36 *frames* para desenhar a ação.

Straight Ahead Action and Pose to Pose - São abordagens diferentes ao estilo de animação tradicional. Straight Ahead Action impõe que as *frames* sejam ilustradas desde do início até o fim. Enquanto Pose to Pose impõe que os key poses da personagem sejam ilustrados primeiro, só depois é que os movimentos entre as *key* poses são ilustrados. Naturalmente cada estilo tem vantagens e desvantagens. Enquanto o Straight Ahead Action cria ações mais dinâmicas, fluida, espontâneas, por exemplo permitindo erros que adicionam personalidade à animação. Enquanto Pose to Pose, a animação pode ser facilmente planeada, controlada de modo a ter um bom *timing*, não é necessário descartar a animação inteira se esta tiver um erro e é capaz de ser avaliada e aprovada com antecedência, porque é possível demonstrar uma ideia genérica da animação através das suas *key* poses.

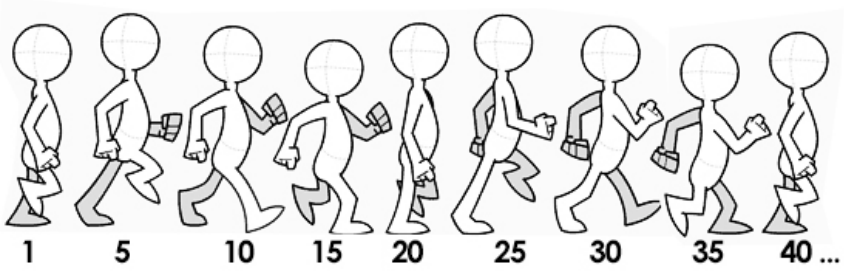


Figura 2 - Exemplo de Pose to Pose

Follow Throught and Overlapping ActionFollow Throught indica que quando um objeto em movimento para bruscamente, partes do seu corpo continuam em movimento antes de pararem completamente. Por exemplo, quando uma pessoa com um cabelo longo para de correr, o seu cabelo continua a mover-se, e só mais tarde fica parado.

Overlapping Action- Relembra que partes num corpo podem ter velocidades diferentes. Por exemplo, enquanto uma personagem gorda está a correr, a sua barriga agita-se a uma velocidade diferente à do corpo.

Arcs - As ações de pessoas, criaturas, e algumas vezes objetos, têm um movimento curvilíneo.

Secondary Action - Enquanto há uma ação principal, esta será suportada, fortificada por uma ação secundaria. Por exemplo, quando a ação principal for chorar, a secundaria pode ser limpar as suas lágrimas com um lençol.

Exaggeration - É a intenção de tornar o movimento da animação mais extrema, exagerada, de modo a ser mais vibrante e viva, como tal as ações das personagens tornam-se mais claras e visíveis ao público.

Solid Drawing - É a importância do conhecimento de conceitos básicos e técnicos necessários para animação, tal como conhecer anatomia dos corpos, conseguir desenhar uma personagem de várias perspetivas, o que era antigamente priorizado nos estudos da Disney. Afinal, conhecer estes conceitos permite uma animação rápida e compreensível.

Appeal - É um princípio importante em animação, a personagem tem que ser apelativa, a audiência tem que ter a capacidade de conectar-se com a personagem. Os movimentos da personagem e o seu design visual devem conseguir representar a sua personalidade. Por exemplo, personagens com cabeças e olhos de maiores proporções em relação ao corpo parecem ser mais jovens e ganham mais simpatia da audiência, ou vilões cujo design tem que os tornar antipáticas à audiência.

"This new art of animation had the power to make the audience actually feel the emotions of a cartoon figure" Thomas F. & Johnston O. (1981) *The illusion of life*. Italy, Walt Disney Productions

Com a evolução da área de animação, naturalmente começaram aparecer projetos grandes e complexos, como tal eram necessários métodos, ferramentas para planear com clareza a animação de modo que houvesse consistência, fluidez e *timing* ao longo do projeto de animação. Tais como:

Storyboard - É a representação visual da história do vídeo, como se fosse uma banda desenhada. Inclui ideias iniciais de ângulos da câmara, esboços iniciais dos *frames* importantes do vídeo, com os seus respetivos segmentos do *script*. Em qualquer projeto de animação, é sempre útil a fabricação de um *storyboard*, não só porque permite uma análise do produto final das primeiras fases do projeto, como também é ideal como um guia para as próximas fases de produção, aumentando a coesão das diversas áreas do projeto (por exemplo: som, diálogo, imagens...)

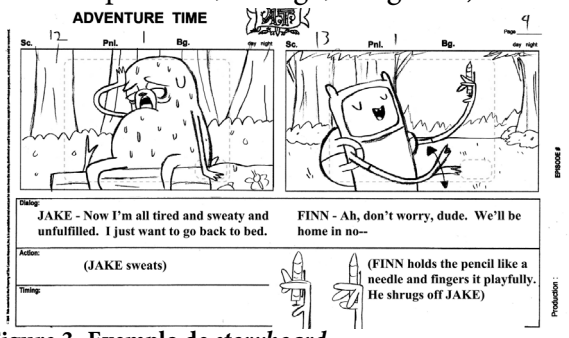


Figura 3- Exemplo de *storyboard*

Animatic - É uma versão animada num *storyboard* em formato de vídeo. Ideal para certificar que ao longo do vídeo a animação tenha um bom fluxo visual, *timing*. Porque normalmente a animação do vídeo é dividida e trabalhada por diferentes artistas, logo esta ferramenta permite averiguar com antecedência a coesão e consistência das várias animações ao longo do vídeo total.

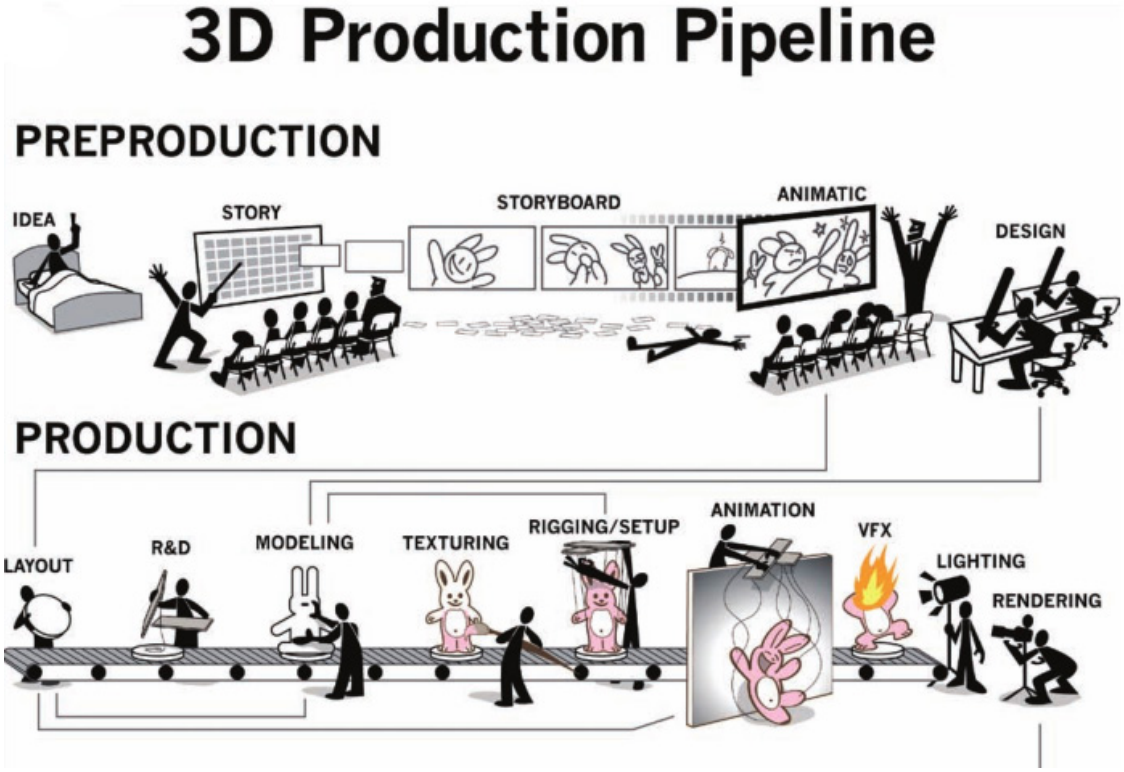
2.2 A Animação no Desenvolvimento de Jogos

Normalmente a indústria costuma produzir projetos 3D com o seguinte modelo de 3D pipeline composto por três fases principais na seguinte ordem: Pré-produção, Produção e Pos-Produção,

A Pré-produção de todos os projetos começam com uma ideia, história, são construídos *scripts* e *storyboards*, testados com um *animatic* e finalmente decide-se o design final. Estes passos são normalmente realizados por uma equipa de artistas, responsáveis pelo projeto e uma equipa de gestão que cria o plano de produção. Esta fase é importante para o sucesso do projeto, quanto melhor for a preparação, definição das ideias, mais facilmente se previne complicações ao longo da pipeline. De tal modo que existem projetos em que metade do seu calendário é ocupado com a Pré-produção.

Na Produção são desenvolvidos os vários elementos do projeto, som, visuais... Se a Pré-produção foi realizada com sucesso, então Produção torna-se mais eficiente, por exemplo seria contra produtivo se modelos, animações, sons tivessem que ser descartados porque houve uma mudança da história ou design. Naturalmente poderá haver mudanças do design do projeto durante a fase de Produção, mas normalmente são incrementos ou pequenas alterações.

A Pós-produção tem como objetivo dar os últimos retoques ao produto final, introduzindo alguns efeitos especiais 2D, corrigir as cores, corrigir erros e ser avaliado. Com o propósito de certificar que o produto está pronto para ser publicado.
Tendo em conta que Produção é a fase mais complexa que cobre diversas áreas, vamos analisar essas áreas e a sua ordem de produção:



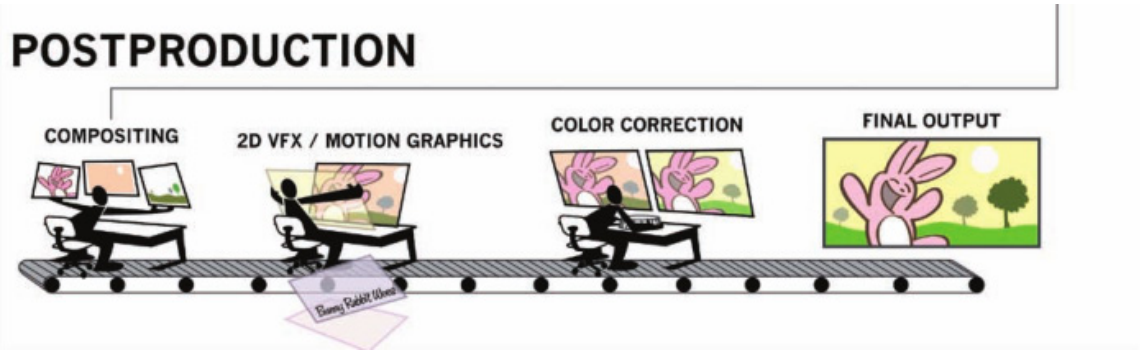


Figura 4- Pipeline 3D

Layout

Criação numa versão 3D do *animatic* com modelos *proxy*. Esta componente é vital porque, ao contraio dos *animatic* 2D, em *animatics* 3D podemos averiguar se os ângulos da câmera a escala das personagens em relação a câmera ou a distância entre objetos convém estarem corretos, porque a sua correção não é fácil. Portanto o 3D *animatic* é utilizado como guia para as escalas entre objetos e serve como base para as diversas equipas do projeto ao longo da fase de Produção.

Rersearch and development (R&D)

Em certos projetos, os artistas podem se defrontar com dificuldades, ou incapacidades em produzir o projeto, porque ou são limitados pelo software, hardware, ou necessitam de um efeito cujas técnicas, métodos desconhecem ou necessitam reproduzir um efeito que nunca foi feito na sua indústria. Estes desafios são abordados em *research and development* com o objetivo de encontrar soluções. R&D é abordado ao longo da pipeline 3D, onde artistas de diversas áreas em conjunção com diretores técnicos, tentam resolver futuros desafios técnicos do projeto. Por exemplo, o jogo Rayman legends tiverem que conceptualizar e produzir software para facilitar a criação de níveis digitais para artistas. Na produção do vídeo Paperman tiveram que criar um software para preservar o traço de artista dos esboços nos modelos 3D. No filme Tangled, a equipa R&D descobriu um método para que a equipa de animadores conseguisse controlar o cabelo da Rapunzel, de modo a que se comportasse e movimentasse como um cabelo real.

"We're interested in taking the artist's original intent with the stroke they lay down on paper and making sure we can faithfully get that into a computer. So we looked at all the drawing tools out there. All of them. All the commercial tools, all kind of under wraps R&D things for drawing—and we were left unsatisfied. A lot of the tools don't faithfully record what the artist. Almost all of our artists found themselves drawing and re-drawing to try to basically beat the computer into submission—beat the line into what they wanted it to be.."
(Hendrickson A. & Whited B. entrevista por Kaganskiy J. Março 5, 2013, recuperado de www.fastcolabs.com/3006276/open-company/trying-woo-animators-disney-accidentally-invents-paperman-method)

Modeling

Modeladores são responsáveis por criar os modelos 3D que se tornam no corpo de personagens, objetos, cenários, que serão utilizados ao longo do Projeto. Modelo 3D podem ser construídos a partir de referências, com um scanner 3D, através de escultura digital ou através da combinação de simples figuras geométricas, tais como esferas, cubos, cilindros, superfícies planas. Mas antes de um modelador começar a modelar, ele tem que primeiro escolher o tipo de geometria ou seja a constituição dos modelos, superfícies de subdivisão, NURBS (non-uniform rational B-splines) ou polígonos.

Existe muita informação para cada um dos tipos de geometria, mas nesta dissertação vamos só focar nos polígonos, porque estes são normalmente os mais utilizados em jogos e por modeladores 3D. Portanto a superfície num objeto 3D é composto por polígonos interligados, designado por Polygon mesh. Cada polígono é composto por 3 ou mais vértices, as arestas, e a face. A forma mais básica num polígono é Tri (3 arestas, triângulo), a mais preferida e usada é a Quad (4 arestas, quadrangulares). Modeladores evitam usar qualquer forma com mais do que 4 arestas. Porque abre a possibilidade a polígonos com reentrâncias, cavidades o que pode causar problemas no projeto, mais especificamente pode facilmente complicar ou inviabilizar algoritmos de cálculo de intersecção, como tal, não têm suporte direto na base computacional.

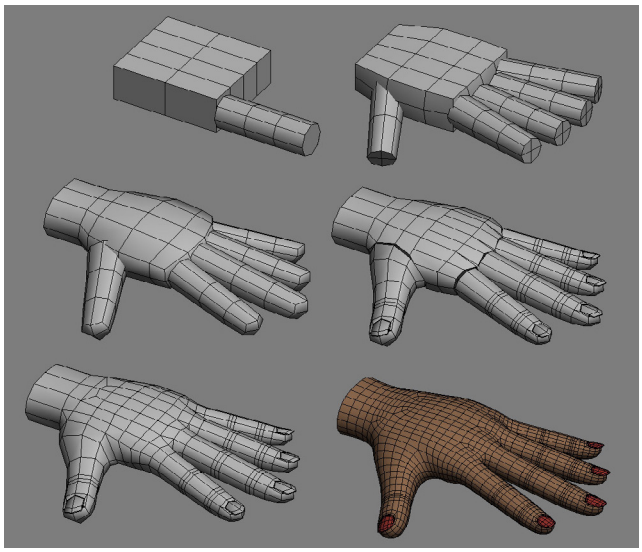


Figura 5- Exemplo de *Polygon mesh* constituido por *Quad*

Topologia é a composição, organização dos polígonos no *mesh*. Um *mesh* com boa topologia pode ser deformada de várias maneiras sem se rasgar ou ficar cortado. Uma das razões porque os polígonos Quad são os mais usados deve-se à composição do seu agrupamento ser ordenado, simples, como uma grelha, A introdução num polígono Tri numa malha de Quad causaria uma deformação, inchaço no mesh (mais proeminente quanto mais redondo for o *mesh*). Uma boa topologia, também facilita o processo de *rigging*, melhora a manipulação da pose do *mesh* de modo a que as deformações do *mesh* sejam exatas, o que é importante na aplicação de textura. Polycount é o número de polígonos num *mesh*, quanto mais polígonos tiver maior o seu detalhe, mas maior poder de processamento é necessário para manipular o *mesh* e maior tempo necessário para o renderizar. Aliás um jogo pode ter modelos com versões de Polycount diferentes, por exemplo modelos com níveis altos de Polycount (grande detalhe) são mais usados em vídeos cinemáticos, enquanto as suas versões de níveis de Polycount mais baixo são ideais para *gameplay*, onde a sua manipulação depende do jogador e o computador precisa renderizar em tempo real. Ou quando um modelo encontra-se longe ou escondidos da câmera, estes não necessitam muito detalhe, polígonos.

Na criação e design dos modelos, principalmente quando se pretende atingir realismo, deve-se ter em atenção com Uncanny valley, que apesar de ser um fenômeno estudado originalmente no campo da robótica, também é aplicável na área 3D. Segundo esta teoria, quando um robô, modelo fica parecido e quase atua como um ser humano mas não exatamente, a audiência sente uma aversão ao modelo, tornando-o desagradável e repulsivo. O que é extremamente problemático em animação, principalmente porque vai contra um dos Twelve Basic Principles of Animation, o principio de Appeal. Este gráfico representa o fenômeno Uncanny valley, ilustrando os níveis de conforto do sujeito com o modelo em relação ao nível de semelhança do modelo com um ser vivo.

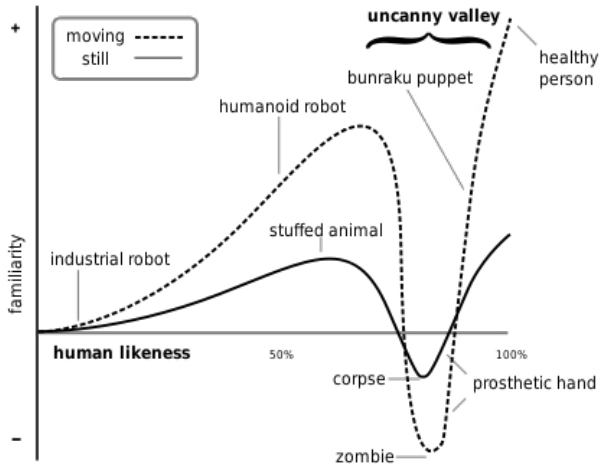


Figura 6- Gráfico do *Uncanny valley*

Texturing

Normalmente os modelos são automaticamente criados com uma cor básica uniforme e neutra (normalmente cinzento), portanto os artistas de textura têm a responsabilidade de transformar a superfície do modelo igual às superfícies representados nos conceitos de arte ou igual à superfície do seu correspondente real.
As texturas ou são criadas a partir da combinação de fotografias, por exemplo padrão de parede de tijolos (através do Adobe Photosop) ou são pintadas à mão, e mais tarde aplicadas nos modelos. Embora com os avanços de software, também é possível pintar as texturas à mão diretamente nos modelos (por exemplo através de Autodesk's Mudbox, Maxon's BodyPaint 3D, ou Pixologic's Zbrush).

Para um artista de textura conseguir aplicar uma textura 2D ao modelo 3D necessita de criar um UV *map*, estes são responsáveis em projetar, embrulhar a textura 2D no modelo 3D. Basicamente atribui-se coordenadas UV / marcadores dos polígonos no modelo e a seguir desembrulha-se o modelo numa imagem 2 dimensões, que servira como um mapa das coordenadas dos polígonos, o UV *map*. Portanto, quando um artista pinta a textura no UV *map*, automaticamente estamos a associar as coordenadas da textura com as coordenadas UV dos polígonos, tornado possível a aplicação da textura do modelo. Este processo é designado por UV *mapping* . Ainda não existe software que consiga criar automaticamente um excelente UV *map* com apenas um clique num botão, normalmente é necessário modificar manualmente o UV *map* gerado pelo computador.

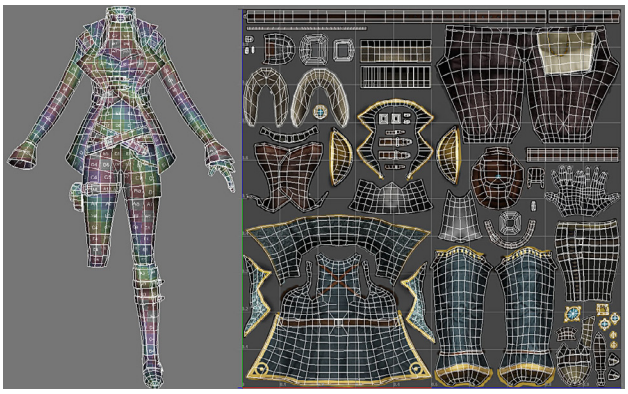


Figura 7- Exemplo de *UV map*

Shaders (também conhecidos por material e *surface*) são instruções implementadas pelo artista de textura na superfície do modelo, em como se deve comportar com o ambiente (o que o rodeia). Alguns atributos básicos de *shaders* são:
Cor- atribuir uma cor ou textura ao modelo.
Ambience - como é que a quantidade de luz de ambiente afeta a superfície do objeto.
Transparência- a quantidade de luz capaz de atravessar pelo objeto, tornando-o transparente, normalmente utilizado para recriar vidro.
Refletivo - o quanto refletivo o objeto é, ou seja a quantidade de luz que a superfície do objeto consegue refletir. Translucidez - Quantidade de luz que atravessa pelo objeto, ideal para recriar papel.
Refração - Este atributo indica se a direção da luz muda quando atravessa pelo objeto, incluindo o ângulo de desvio, por exemplo o efeito de quando se olha através num copo com água e os objeto estão distorcidos.
Brilho - simula o efeito de auto iluminação, por exemplo a luz emitida pelo ecrã num monitor de computador. Incandescência - determina a qualidade de auto iluminação.
Realce especular (*Speclar highlight*) - cria manchas brilhantes ao longo da superfície do objeto, que na vida real corresponde à reflexão da fonte de luz .
Colisão (*Bump*) - simula textura ao longo da superfície através de *bumps maps*. São imagens monocromáticas que representam mudanças ao longo da superfície, a cor preto representam declives, descensões na superfície e a cor branco representam elevações, ascensões na superfície. Mas o *bump map* não altera a geometria do modelo, altera as normais da superfície do modelo e consequentemente como esta reage à luz, criando a ilusão de elevações e declives ao longo da sua superfície. Por exemplo consegue simular uma superfície rugosa.

Existem vários tipos de *shaders*, por exemplo:
Flat shaders - um dos *shaders* mais básicos, cujo resultado é sempre uma cor uniforme, sem gradientes (exemplo de uso: imagem de fundo do céu).
Lambert - com o mais básico de iluminação sem efeitos de reflação e Realce especular, indicado para materiais que não brilham muito, papel, madeira inacabada.
Blinn - permite reflexão e Realce especular, próprio para plásticos, metais, cabedal.
Phong - cria um aspeto brilhante, com um Realce especular intenso, próprio para plástico.

Assim que o artista de textura configurar os atributos do *shader* num modelo, precisa introduzir a informação visual nos atributos através da atribuição mapas, tais como:
Bump maps. Mapas de cor- com a informação da distribuição das cores ao longo da superfície no modelo.
Specular maps - Afeta o comportamento de Realce especular (*Speclar highlight*) ao longo da superfície, criando a ilusão de arranhões, perturbações.
Mapas de transparência - Cria um gradiente de transparência ao longo da superfície, bons para recriar vidros sujos, gelado ou colorido.

Mapas de reflexão- Controla o degrau de reflexão ao longo da superfície.
Displacement maps - Depois de ser aplicado no modelo e renderizado, altera a geometria da superfície do modelo de modo a criar uma nova forma.
Ambient occlusion map- São imagens monocromáticas que criam sombras suaves, desfocadas ao longo da superfície do modelo
Normal maps - Tal como *bump maps*, cria a ilusão de textura, mas em vez de se basear numa imagem de tons de cinza, utiliza uma imagem com as cores verde, azul e vermelho (RGB). Tipicamente utilizado em jogos.

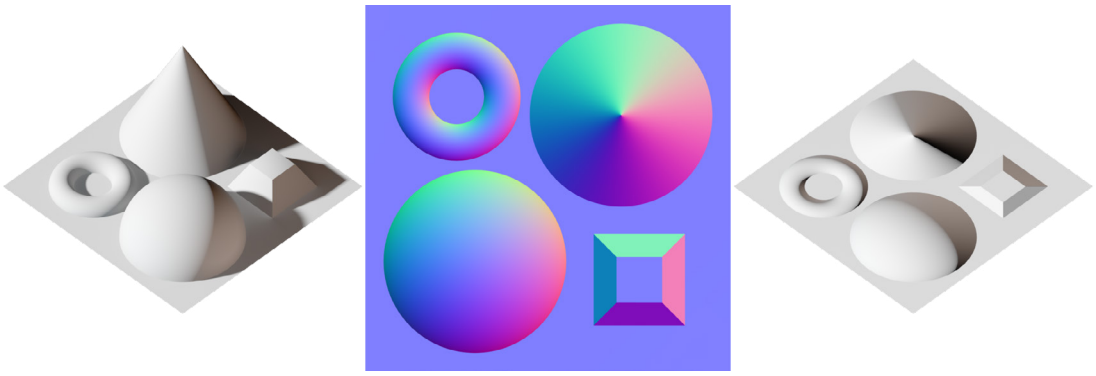


Figura 8- Exemplo de *Normal map*

Rigging/setup

Rigging é responsável pela criação e implementação de um *rig* (também conhecido por esqueleto), num modelo, permitindo a sua manipulação, controlo e animação, do modo mais rápido e eficiente possível. O artista de *rigging* implementa um esqueleto, composto por *joints*/articulações, *bones*/ossos. A seguir cria controlos que permite aos animadores animar o modelo, através da translação e rotações dos joints. Depois usam *deformers* para conectar os joints e bones do rig com a geometria do modelo, de modo a este acompanhar com os movimentos dos *rigs* e finalmente para garantir o realismo do movimento do modelo implementa-se mais *deformers*.

A implementação desta área no projeto é bastante complexa e normalmente o artista de *rigging* necessita ser acompanhado com a equipa de R&D.
Deformers são responsáveis pela ligação entre o modelo e o *rig*. Por exemplo:
Skining/Binding é um *deformer* que atribui valores à cada vértice do modelo que especifica o quanto são afetados pela articulação do joint.
Constraints permitem criar um sistema em que certos objetos partilham o mesmo tipo de movimento. Os *constrains* mais utilizados são:
Point- A translação do objeto x é partilhado com o objeto y
Aim- A rotação do objeto x segundo um alvo é partilhado com objeto y, ideal para um sistema de controlo de olhos
Orient- A rotação do objeto x é partilhado com objeto y
Scale- A mudança de escala do objeto x é partilhado com o objeto y

Todos os *rig* são sistemas com hierarquia, de modo a definir como o movimentos partes do modelo influenciam outras partes. O mais básico é hierarquia Parent/child, em que categoriza bones do *rig* por parent ou child, ou seja, um bone *child*, pode mover, rodar sem afetar o bone *parent*, mas se o bone *parent* mover então o bone *child* acompanha com o seu movimento. Esta relação chama-se *parenting*. É possível haver vários *child* com o mesmo *parent*, chamados *siblings*. Embora que exista aplicações que conseguem customizar a relação *parenting*.

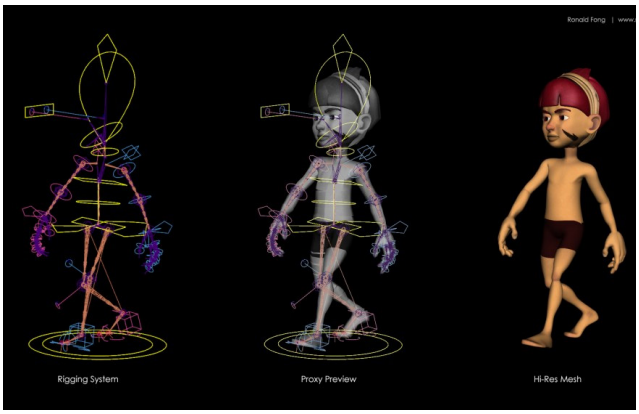


Figura 9- Exemplo de um *rig*

Kinematics permite estabelecer a ordem na hierarquia do *rig*, basicamente fornece aos animadores duas diferentes abordagens de criar uma animação. A Forward Kinematics (FK) estabelece a ordem de hierarquia predefinida do *rig* e Inverse Kinematics (IK) inverte a ordem de hierarquia predefinida do *rig*, portanto se a ultima joint na hierarquia mover-se, o resto dos joints, bones acompanham com o movimento. Normalmente o IK é excelente para animar pessoas a andar, por exemplo, um pé ao tomar um passo permanece na nova localização e consequentemente a anca acompanha. Inverse Kinematics é mais rápido que Forward Kinematics, mas ambos são úteis para alcançar um bom movimento natural.
Normalmente os *rigs* são baseados à partir do esqueleto humano, animal, como tal é conveniente que os artistas de textura tenham no minimo um conhecimento básico da anatomia do corpo que pretende simular. Contudo, deve-se ter em conta que a anatomia de todos os corpos têm limites, uma pessoa não consegue rodar a sua cara 180º. Como tal é necessário implementar limites na rotação dos joints. Adicionar estes limites é fundamental para usar inverse kinematics.

Animação: análise comparativa do cinema com os jogos

"poor animation can kill any 3d project. Even if the models are perfect and the lighting is great, unrealistic or distracting motion will pull your audience out of watching your project"
Beane A. (2012). 3D animation essentials

Muitas das técnicas, ferramentas de animação tradicional foram adaptadas para animação 3D. Mas ao longo do tempo a área de animação 3D tem evoluído, amadurecido, criando a sua própria linguagem e ferramentas. Como tal um animador tradicional sempre conseguiu transferir-se para a indústria de animação 3D, mas ao longo do tempo a curva de aprendizagem para animação 3D tem aumentando.

Os 12 princípios básicos de animação também foram influenciados pela indústria 3D, digital: O principio de Solid Drawing requer que um animador 3D deva ter um boa compreensão do seu software de animação 3D, mas não significa que não precisa saber desenhar, afinal animação 3D continua a ser uma comunicação visual, e a maneira mais rápida de demostrar o produto final é a partir de desenhos. Tendo em conta ao principio de Straight Ahead Action and Pose to Pose, especificamente Pose to Pose. Em vez de ilustrar todos os *frames* na animação entre *key frames*, a animação, *frames* pode ser calculados e produzidos por computadores.

O objetivo de um animador não é criar uma obra de arte mas criar movimento, como tal é aceitável deformar o modelo em ordem a obter um movimento fluido. Estas deformações podem ser por exemplo:

Breaking body- alterar a composição do modelo de tal modo que pareça que estamos a deslocar/ partir o corpo do modelo.

Smear- substituir partes do corpo do modelo por borrões, de modo a representar a trajetórias de movimento demasiado rápidos ao olhos da audiência.

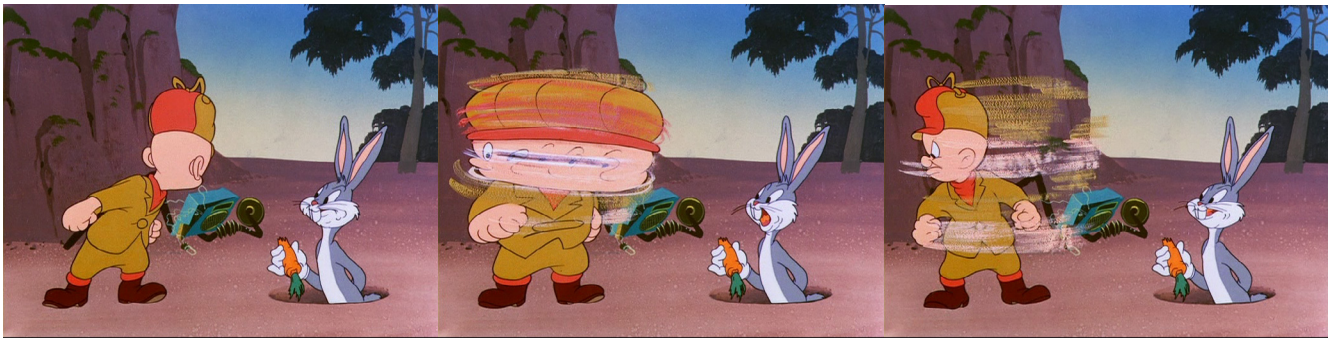


Figura 10- Exemplo de Smear

"Remember your foundations as an artist but don´t be afraid to break a few bones if it´s gonna help your motion look better, remember you're creating movement, not individual pieces of art"
Mariel Cartwright (2016) *Making Fluid and Powerful Animations For Skullgirls* from https://www.youtube.com/watch?v=Mw0h9WmBlsw

Em animação cinemática, o animador controla a posição e direção da câmera, conhece a história do filme e possui controlo total das personagens, de modo a que consigam criar e refinar animações com personalidade, apelo visual, emocional. Mas em jogos, os animadores têm de ter em consideração a presença de um jogador e das ferramentas, mecânicas do jogo que são afetadas ou afetam a animação.

Ao contrário dos filmes em que a animação tem que ser visualmente agradável e parecer natural, em jogos prioriza-se que o jogador se sinta bem com a animação. O que envolve que o jogador sinta que a animação seja "*responsive*", flexível, instantânea às suas intenções. No entretanto, o jogador necessita que as animações dos *npc* sejam fáceis de ler, de analisar, de modo a que ele consiga rapidamente prever e reagir à ação do *npc*.

Utilizando a animação de um golpe como exemplo, podemos estudar as diferenças entre um soco num *npc* em relação ao soco do protagonista, a personagem controlada pelo jogador. Antecipação do soco do Protagonista, os movimentos do corpo necessários para formar o soco, tem que ser quase instantânea, de forma a que o soco coincida com o momento preciso em que o jogador queria dar o soco. Mas isto torna a animação muito rápida e difícil de ser processada pelo jogador, como tal usa-se o princípio de "follow through" e *smear* na animação para completar a informação do movimento ao jogador. Enquanto que a Antecipação do soco do *npc* deve ser suficientemente longa de modo a que seja possível com que o jogador consiga notar a ação e prever que é um soco.

Tendo em conta que jogadores são de tal modo sensíveis à animação, é importante garantir que eles sentem peso, impacto dos movimentos das personagens. Isto pode ser realizado por estendendo o momento de *hitspot* e introduzindo *overshoot* na animação

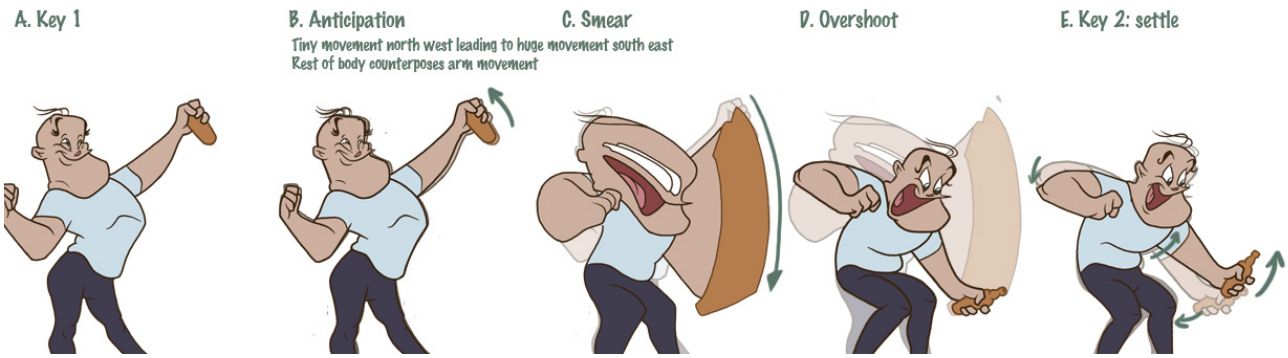


Figura 11 - Exemplo com hitspot e overshoot

Motion-Capture Animation é um novo processo, sistema de capturar a posição e os movimentos num ator. O sistema processa e transforma os dados do movimento em dados de animação 3D, permitindo aos *rigs* dos modelos 3D replicar os movimentos do ator. Tipicamente para captura a data de movimento, os atores usam fatos colados à pele com marcadores ao longo do corpo, o computador consegue capturar a posição dos marcadores à partir de varias câmeras durante a performance e consequentemente deduzir, calcular a data de movimento

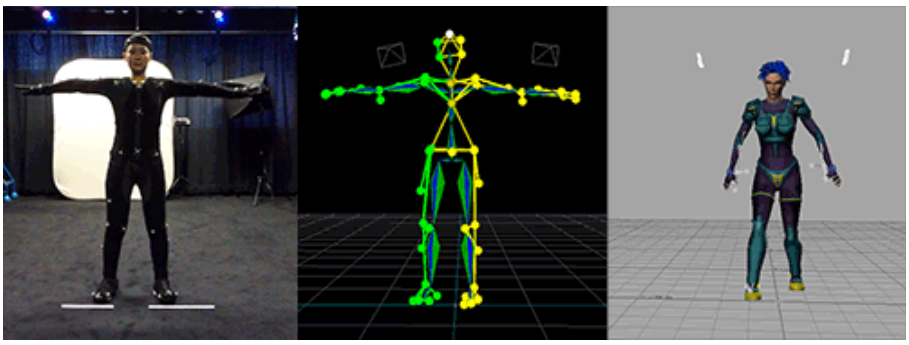


Figura 12 - Exemplo de Motion-Capture Animation

Efeitos visuais 3D (VFX)

O artista de efeitos especiais 3D é responsável pela animação de tudo exceto as personagens e os *props* com que interagem, por exemplo cabelo, tecidos, pó, água e fogo. Normalmente estes efeitos especiais dependentes de software com motores de física que simulam ar, gravidade e vento, porque quanto mais natural e complexo for o seu aspeto e movimento, mais difícil é de os animar à mão. Portanto é necessário com que o artista de efeitos especiais 3D tenha um conhecimento básico de físicas e matemática e as consiga fundir com o seu senso artístico. Mas tem que ter sempre em conta que o seu objetivo final é com que os efeitos melhoram o cenário, em vez de roubarem involuntariamente a atenção do público.

A simulação dos efeitos 3D produz muitos dados que precisam ser calculados e processados. Portanto, não é fácil manipulá-los e exigem uma alta carga de trabalho ao processador. Os efeitos especiais 3D podem se dividir nas seguintes especializações:
Particles - Um emissor, posicionado no espaço pelo artista, liberta um fluxo de pontos, partículas, cujo movimento respeita as leis de física, gravidade, vento, fricção. Ideal para simular, poeira, fogo, chuva, neve ou enxames.
Hair and Fur - São sistemas que criam cabelo e pelos sujeitos a perturbações, que se comportam de

modo fluído e realista. Embora que este sistema também pode ser utilizado para criar antenas, caudas, tentáculos, é um dos sistemas mais difíceis de controlar e cuja renderização requer muito tempo, sendo difícil obter resultados realistas.
Fluids - São simulações especiais de partículas que conseguem simular o movimento de fluidos através de uma equação. Não só restrito à simulação de líquidos, como também ao fumo, fogo e plasma.
Rigid Bodies - São modelos que representam sólidos, que ao colidir com outros objetos não deformam, cujo movimento é condicionado pelas leis da física. Normalmente utilizado para colisão básica de objeto rígidos, fracionamento de objetos, por exemplo a simulação de cacos de vidro a cair.
Soft Bodies - São modelos que representam sólidos, que devido à colisão com outros objetos, deformam-se. Estes tipos de simulações podem exercer muita carga de trabalho ao computar. São normalmente utilizadas para criar tecido realístico, músculo, cartoon *hair*.

Lighting

O artista de Iluminação, simula os vários tipos de luz do mundo real num cenário 3D, por exemplo *spotlights*, luz de lâmpadas, luz do sol... Portanto a função do artista de Iluminação é o posicionamento, direção e controlo dos atributos das luzes (Intensidade, color, decadência), também pode incluir a manipulação das propriedades das sombras do objetos de modo a condizer com a intensidade das luzes de modo alcançar o cenário, ambiente predeterminado.
Spotlight - Emite uma luz numa direção e localização específica.
Omni/Points Lights - Emite luz numa localização em todas as direções
Infinite/Directional lights - Emite raios de luz que são paralelos entre si, produzindo sombras paralelas entre si criando a sensação que a fonte luz está muito longe. Ideal para simular a luz do Sol ou da Lua.
Ambient Light - Estas luzes são usadas para simular iluminação global
Area light - Uma das luzes mais complexas, em vez de emitir a luz num ponto de localização, emite numa área, por exemplo a área numa janela ou ecrã num monitor.

Rendering

O passo final da fase de produção, que combina os modelos, *rigs*, animações, *shaders*, texturas efeitos especiais 3D e iluminação e os transforma, renderiza-os em imagens.
Dois métodos básicos de renderização são:
Scaline - Renderização mais rápida, consequentemente esta não calcula reflexões, refrações ou sistemas complexos de iluminação global. Sendo ideal para renderização de cenários *cell-shaded*, parecido com cartoons, mas principalmente para a pré-visualizações do cenário ao longo do projeto para observar o seu progresso.
Raytracing - É o método de renderização mais completo, renderiza reflexões, refrações e sistemas complexos de iluminação global, consequentemente o que exerce mais carga ao computador.

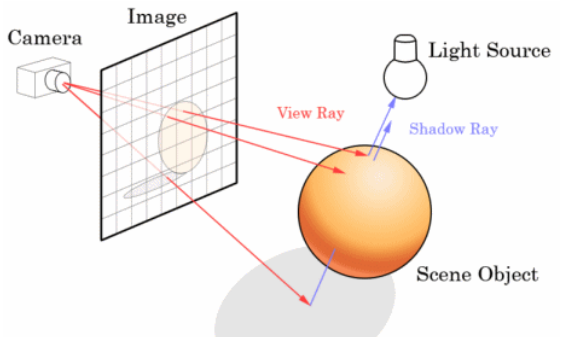


Figura 13- Exemplo de Raytracing

Naturalmente nos jogos, o cenário tem que reagir às ações dos jogadores e estas ações e as suas consequências necessitem ser renderizadas em tempo real. Um método que para obter renderização em tempo real é baking texture *maps*, o que envolve a incorporação da informação de sombras e luzes no mapa de textura. Pois a renderização da reflexão de luz não é fácil.
Apesar de Raytracing renderizar imagens muito realistas, este é extremamente difícil aplicar em renderização em tempo real, pois é um processo complexo que exerce muita carga de trabalho e como tal necessita de muito tempo para renderizar.
Os *engines* de jogos que nos permite jogar jogos de vídeo tais como Unreal e unity engine, também são bem capazes de renderizar em tempo real. Mas tipicamente os designers de jogos preferem implementar os programas de renderização DirectX e OpenGL, porque tem algoritmos rápidos de Scaline e são suportados por placas gráficas .

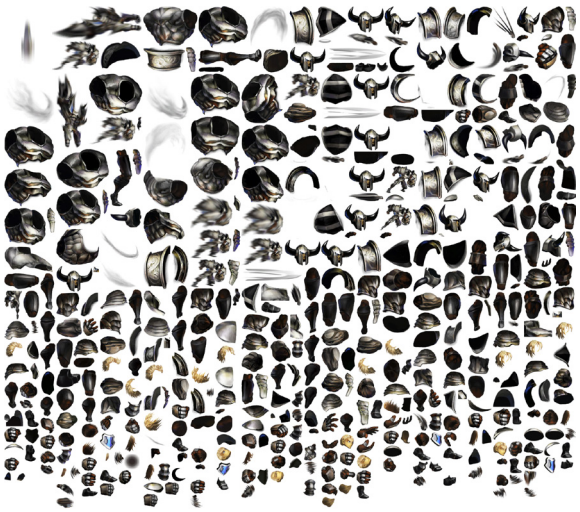
2.3 Técnicas de animação em jogos

No âmbito desta dissertação, fiz a pesquisa e análise de vários jogos com animações interessantes, básicas ou incomuns, que evidenciam estilos de estética e design que influencia a sua animação, tal como, a ilusão de volume em imagens 2D. A partir de entrevistas, palestras dos seus desenvolvedores consegui recolher as técnicas de animação e r os seus pontos fortes e fracos.

Animação modular/ bone animation

Esta técnica envolve a animação de *sprites* como se fossem marionetas, ou seja as personagens são compostas por muitos bocados de imagens, cujo movimento depende de translações, rotações e transformações dos bocados. Possibilita igualmente, inúmeras e diversas animações com um número pequeno, limitado de fragmentos de imagens, o que exige menor memória do disco rígido. Normalmente esta animação é utilizada em jogos com uma câmera fixa, ou seja um jogo "Side Scroller". Se o jogo tiver uma câmera com mais liberdade, o jogador terá acesso a mais ângulos de perspetiva das personagens e por cada novo ângulo é necessário um novo perfil da personagem, o que envolve a produção de mais fragmentos de imagens para o novo perfil. Tipicamente em animações de *sprites* com pequeno orçamento, os animadores têm que ilustrar rapidamente um quantidade imensa de *frames*, para tal ser possível convém que os desenhos sejam simples ou de menor qualidade. Em contrapartida a animação modular permite animar ilustrações ricas em pormenor de alta qualidade.

Exemplos: Rayman Legends, Muramasa e Dragon's Crown



Animação desenhada à mão

Envolve o uso de animação tradicional em jogos, ou seja a animação das personagens e cenário do jogo são totalmente concebidos através de desenho à mão. Normalmente é utilizado por animadores, equipas com experiência e paixão por animação 2D, animação de *sprites*. As animações são mais fluídas e suaves porque são mais fáceis de implementar os 12 princípios de animação, por exemplo o "Squash or Stretch ". Todavia, apesar de ser um processo simples, requer muito trabalho e tempo.

"Alex: each character has essentially 8 animations each, filmed from two directions, northeast and southeast. With each animation coming in between 30-60 frames, that means each character class can have up to 1000 unique frames of animation"

"Finally, making a finished animation in 2D can be time-intensive so we always start prototyping with rough animations. It's much quicker to iterate on programming and game design elements this way " E3_2011 (2011, Junho 11) Supergiant Games' Bastion. recuperado em http://www.cgsociety.org/index.php/CGSFeatures/CGSFeatureSpecial/supergiant_games_bastion

Exemplos: Jotun, Banner saga, Cuphead e Castle crashers.

Animação de *sprites* a partir de modelos 3D

Esta técnica baseia-se em que as personagens, objetos e cenário são criadas e animadas em 3D, as animação são exportadas em series de imagens e mais tarde são agrupadas em sequência pelo game *engine* 2D para criar as animações. Animação de *sprites* a partir de modelos 3D tanto pode ser ideal para equipas pequenas e sem experiencia em 3D game *engines*, como também pode ajudar a capturar o estilo 2D na animação a partir de modelos 3D.
"The reason why they went this route was to keep the whole game running within a 2D engine and to avoid developing the game with a 3D engine which often proves to be a gigantic chore for game developers that don't have an incredibly huge staff." (2015, Abril 21) GuiltyGearXrd's Art Style : The X Factor Between 2D and 3D [video], GDC recuperado em https://youtu.be/yhGjCzxJV3E?t=1787

Contudo, sempre que for necessário fazer uma modificação à animação, à cor da personagem, é necessário modificar a animação 3D e voltar a renderizar e implementar no jogo.

Exemplos: Bastion, Transitor, GuiltyGearXrd

Efeitos especiais 2D

Uso de animação de *sprites*, num espaço 3D, para representar efeitos especiais, poeira, raios, fogo. Ajuda alcançar o estilo 2D, "cartoon" e não exerce muita carga de trabalho no processador, ao contrário da alternativa efeitos especiais 3D. Mas em certos jogos cuja câmera tem muita liberdade, os *sprites* podem ser facilmente reconhecidas como imagens bidimensionais, arruinando a experiencia de profundidade, portanto nessas situações convêm que os efeitos sejam pequenos (por exemplo: nuvens de poeira) ou rápidos (por exemplo: flash de luzes)

Exemplos: Wildstar - Free-to-Play Trailer, League of Legends Music: Get Jinxed

Cel shading

É um tipo de renderização que torna os modelos 3D mais 2D, com estilo de cartoon, banda desenhada. A partir de cálculos matemáticos entre as normais dos modelos e a fonte de luz, os meios-tons, sombras e gradientes dos modelos são respetivamente substituídas por cores únicas . Mas sem controlo e alterações manuais nas normais dos modelos, este processo pode reproduzir resultados diferentes do desejado.

Exemplos: GuiltyGearXrd, zelda wind waker

Iluminação num jogo 2D (*normal map*)

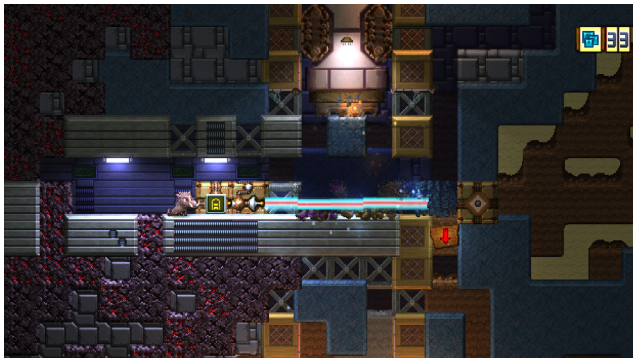
Em vez de aplicar *normal map* em modelos 3D, aplica-se em sprites, imagens de cenário, de modo a estes serem afetados pela luz e sombras criando a ilusão de profundidade e volume.

Habitualmente, *normal maps* são gerados pelo computador a partir de um modelo 3D e mais tarde são modificados manualmente. Mas no caso desta técnica não há modelos 3D, portanto os *normal maps* têm que ser sempre criados manualmente.

Existem vários processos para obter *normal map* a partir de imagens 2D, através de *plugins*, por exemplo NVidia Photoshop Plugin, ou criando 4 imagens, em que cada uma tem a personagem com sombras em que a luz está à sua esquerda, em cima, à sua direita e em baixo. Através do programa Sprite Lamp, ou seguindo o tutorial Normal Map Photography de Ryan Clark, as 4 imagens transformam-se no normal map.

A desvantagem deste método é que para criar manualmente *normal maps* com resultados excelentes, ou realistas, é necessário muita experiencia e treino.

Exemplos: Full Bore



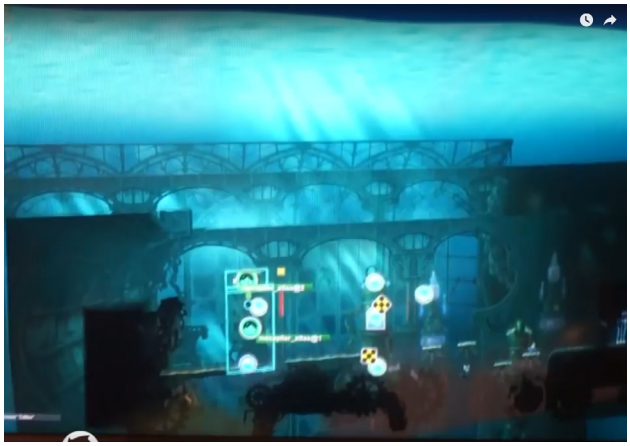
MultiPlane Camera effect

Método inventado pela animadora alemã Lotte Reininger, e uma década mais tarde explorada por Bill Garity, técnico dos estúdios Walt Disney, no desafio em incorporar volume, profundidade nos seus filmes de animação 2D.

"Imagine you're traveling by car on a long road trip. The road stretches out ahead of you into a wide -open landscape. In the distance you see a range of tall, hazy mountains. As you drive, you see trees and telephone poles whiz past you. At first, they appear tiny in the distance. Gradually, they get larger as you travel closer to them and pass them. Strangely, the mountains still appear to be very far away and hazy. They don't appear to have gotten larger despite being closer to them the way the trees and the telephone poles did. Why? Because the mountains are so large and far away, the change in their apparent size is negligible." Multiplane educator guide, recuperado em http://www.waltdisney.com/sites/default/files/MultiplaneGuideCurriculumPacket_Final.pdf

O MultiPlane Camera normalmente cria este efeito dividindo o cenário em três diferentes planos, o *foreground*, *middle ground*, e *background*. Alterando a posição deste níveis relativamente a câmera, é possível criar a sensação de profundidade. Por exemplo, aproximando o *foreground* e *middle ground* em direção a câmera, enquanto o *background* permanece na mesma posição, cria-se a ilusão de que estamos a atravessar o cenário.

Exemplos: Rayman Legends



Processo de Paperman.

Paperman é um filme curto com um novo processo inovador, depois da renderização da animação digital, com o uso de um software customizado, o artista desenha à mão sobre os modelos digitais nas *key frames*. O software anexa os desenhos aos modelos e produz a sua animação entre as *key frames*. Transmite o traço artístico na animação 3D, sendo um excelente processo com resultados híbridos de animação 3D com animação tradicional. Mas é um processo que ainda está em desenvolvimento, pois a equipa responsável ainda pretende melhorar a animação dos desenhos produzidos entre *key frames*. Em conclusão, este método permite modificar as textura e anima essas modificações segundo a animação já preexistente, personalizando o estilo de representação.

Exemplos: Paperman

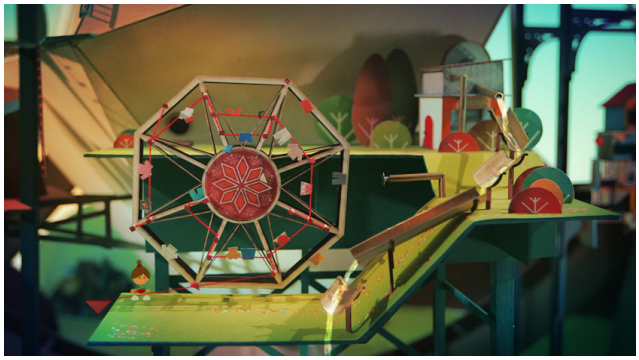
Outline e *inline 3D*

Reprodução de traços *outline* e *inline* típicos de ilustrações 2D em modelos 3D Os traços outline são os traços à volta da silhueta do modelo, através do método Inverted Hull Os traços inline são os traços na superfície do modelo, normalmente utilizados em 2D para indicar os contornos, bordas que se encontram dentro da silhueta do modelo.

Exemplos: GuiltyGearXrd, Bordelands 2

Uso de cenário físico.

Criação física num set, capturado, filmado por câmera é introduzida como espaço, cenário no jogo. Apesar de facilitar e simplificar a construção do espaço, cenário do jogo, principalmente para equipas sem experiência na criação digital do mundo e a suas interações, são necessárias extensas horas de trabalho manual e filmagem para capturar todos os ângulos necessários do set.



3 Abordagem Metodológica

Neste capítulo estabelecemos os desafios a ser explorados para alcançar o objetivo desta dissertação, os processos técnicos a ensaiar para resolver os desafios, de modo a produzir a prova de conceito e por fim a sua calendarização.

3.1 -Desafios

Tendo em conta que os objectivos desta dissertação é a pesquisa de técnicas que funde 2D com 3D, cuja combinação promove profundidade, volume e toque do artista na animação e aspecto gráfico em jogos. O qual pretendo implementar num protótipo do Jogo como prova de conceito. Consequentemente permitindo a avaliação do resultado da combinação de modo a determinar os ajustamento que necessita. No âmbito desta dissertação serei confrontado com vários desafios para executar o procedimento do 2D com 3D no jogo, tais como:

- Explorar como é que espontâneas fontes de iluminação 3D podem afetar imagens 2D sem estas se tornarem planas, com renderização em tempo real.
- Investigar com experimentação de métodos, formas de dar um toque artístico à textura dos modelos 3D.
- Descobrir os limites no jogo do uso de animações complexas.
- Explorar o uso de efeitos especiais 3D mas com estilo 2D, com a gratificação da diminuição de complexidade dos efeitos especiais 3D.
- Salvaguardar a eficiência do jogo, ou seja, manter um equilíbrio entre carga de trabalho do processador e o uso de memória do disco rígido, ao longo do processo de resolução dos desafios anteriormente mencionados.
- Definir as minhas limitações, permitindo determinar o potencial do uso num game *engine* 3D ou 2D, de modo a escolher o mais qualificado segundo a natureza desta dissertação.

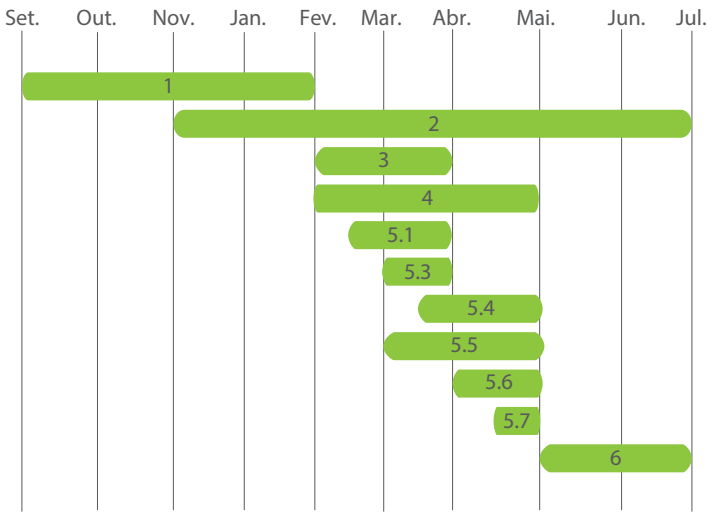
Estes desafios e questões serão explorados nos vários processos da criação do produto da dissertação, o jogo.

3.2 Processo para produzir a prova de conceito

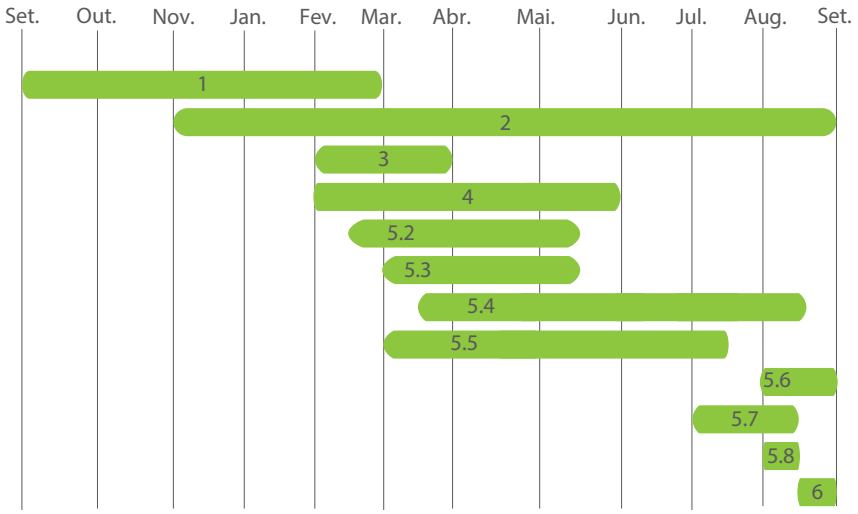
Os processos adotados na realização da prova de conceito passaram por uma pesquisa sobre a opinião pública, conceitos básicos, técnicas e exemplos de animação 3D, 2D, e a sua combinação, incluindo uma investigação em como esta informação pode ser utilizada na criação de um protótipo de um jogo. Mais tarde, a informação foi analisada na escrita da dissertação.
Para criar o nível do jogo foi necessário uma familiarização com o software 3D, através de tutoriais colecionados ao longo da pesquisa, e através da experimentação. Este também envolveu a reprodução das técnicas de animação, de modo a determinar o seu potencial.
Para uma melhor gestão do processo de produção do nível do jogo foi fundamental determinar os limites da minha capacidade de produção.
O processo de testes de usabilidade é responsável pela exposição do nível do jogo a *test users*, de forma a registar a sua experiencia, *bugs* e opinião sobre a estética do jogo, incluindo a animação, com o propósito de analisar os resultados dos testes, determinando o sucesso das técnicas implementadas e consequentemente fazer as alterações necessárias.

3.3 Plano e calendarização

1º Calendário de trabalho



Calendário de trabalho



1- Pesquisa para dissertação
2- Escrita da dissertação
3-Familiarização com software
4-Experimentação de Técnicas
5- Criação do jogo
5.1- Modelação
5.2- Desenho
5.3- Rigging
5.4- Animação
5.5- Programação
5.6- VFX
5.7- Iluminação
5.8- Áudio
6- Teste de usabilidade

4. Conceptualização e Proposta de Game Design

Este capítulo representa a primeira fase do planeamento do desenvolvimento do projeto prático desta dissertação, nomeadamente do desenvolvimento do nível do jogo, e, estabelece a premissa, mecânicas de *gameplay,* design do jogo, o modelo de animação e as técnicas de animação que se pretenderam implementar.

4.1 -Conceito proposto

O gênero deste jogo é de Fantasia, com localização e data indefinidas, inspiradas pelas terras célticas na idade do ferro. O ser humano tem acesso à magia da natureza e de deuses a partir de runas. Esta arte é comum e praticada por druidas. As runas podem estar inscritas em armas, armaduras ou livros, mas a sua inscrição necessita de tempo, materiais e técnicas especificas, tornando a sua preparação essencial para combates o que introduz uma componente interessante no *gameplay*. A personagem principal do jogo adquiriu acesso à magia sem ser a partir de runas, *Witchcraft*. Esta habilidade é anormal, inatural, o seu abuso pode corromper a natureza (por exemplo: dias mais escuros) e atrair atenção e castigos dos deuses (por exemplo, dano ou maldições à personagem). A personagem principal, Witch, no nível do jogo ao explorar uma floresta, vai ter necessidade de travar um combate com bandidos ou cultistas no descampado na proximidade de um castro.

4.2 -Proposta de Game Design

Intencionasse implementar do jogo a alteração do mundo durante o uso de Witchcraft em combate. O mundo escurece drasticamente e os seres vivos tornam-se em sombras cobertos por símbolos, que representam o seus espíritos e pistas sobre as habilidades dos seres vivos.

Estas animações serão criadas tendo em consideração que câmera do jogo vai tem uma vista da terceira pessoa a partir do céu.
Idealmente o sistema de combate, permitia total controlo das personagem, *hack and slash* game, por exemplo: Diablo 3, mas tendo em conta ao minha experiencia e tempo limite para a concretização do protótipo do jogo, o sistema do jogo vai ser dividido por turnos, *turn based* game, por exemplo: Final Fantasy Tactics, Disgaea.

Disgaea


Diablo 3

4.3 -Técnicas a ensaiar

Para a seleção de animações teve-se em consideração as animações básicas, necessárias para um jogo com mecânicas de combate e animações complexas, que não só o tornam mais interessante, mas também lhe confere a capacidade de demonstrar o resultado da fusão de 2D com 3D, por exemplo animações com efeitos especiais.

As animações básicas são: Movimento, o que envolve no mínimo animar a personagem em 3 perspetivas, andar para norte, para sul e andar para este ou oeste (estas perspetivas partilham a mesma animação). Ataque simples (por exemplo: soco ou ataque com espada). Sofrer ferida. Cair inconscientemente/morrer. Estar em pé, pois nenhum ser vivo consegue estar completamente parado, por isso idealmente quando este não está a praticar nenhuma ação deve ter ligeiros movimentos. Esta animação aumenta a ilusão de vida da personagem. Animações complexas: Ataques com elementos (por exemplo, espada com eletricidade) *Dash attack*, ataques físicos com alta mobilidade, magia de evocar criaturas, magia de projéteis (por exemplo: bolas de fogo, picos gelo) e magia laser.

É um jogo 3D *non photorealistic*, apesar de não possuir os recursos necessários para produzir um jogo *photorealistic*, o estilo cartoon de *non photorealistic* combina melhor com os efeitos 2D, oferecendo um contraste de estética interessante.

As técnicas que se pretendem implementar no jogo são multiplane camera, Iluminação num jogo 2D e efeitos especiais 2D, em caso de necessidade devido a problemas com animação e compensações pode-se recorrer à animação modular, cell shading e animação de *sprites* a partir de modelos 3D. O Game engine (motor de jogo) para correr o jogo é Unity, sendo o engine em que possuo mais experiência. Mas os programas que tenciono utilizar para as várias fases da construção do jogo são:-

Fases	**programas**
Game engine (motor de jogo)	Unity
Modelação	Z-brush, Autodesk 3ds Max ou Cinema 4d
Alterar topologia de modelos	Autodesk 3ds Max
baking e texturas	photoshop, Autodesk 3ds Max e Mudbox
Rigging	Autodesk 3ds Max ou Cinema 4d
Criar animação	Autodesk 3ds Max ou Cinema 4d

5. -Desenvolvimento do Protótipo

Neste capítulo serão descritas as várias fases do processo de criação do protótipo do jogo, incluindo as funções que foram implementadas, escolha de tecnologias e como as fases se influenciaram mutuamente.

O primeiro subcapítulo incide na primeira fase do processo, a experimentação das tecnologias e técnicas e como estas induziram às modificações da proposta de design. O Segundo subcapítulo contém o design visual do jogo e a sua evolução. No terceiro, abordamos a produção das personagens, o que envolve o seu desenho, a construção dos seus *rig* e a sua mútua relação. No quarto designaram-se as várias animações implementadas no jogo, incluindo o seu design, amostras das animações e o mapa das suas relações com as outras animações. No quinto, referimos a iluminação, materiais e *shaders* implementados no jogo. No sexto, explicamos o comportamento e constituição da câmera de jogo e descrevemos o comportamento e a constituição da câmera de jogo. No sétimo, explicamos em programação as várias funções que foram implementadas e as que não foram, e, no oitavo, mencionamos os vários sons implementados.

5.1 -Fase de experimentação

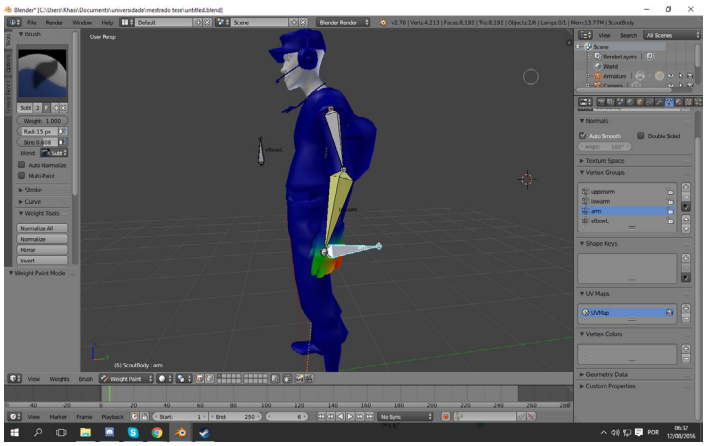
Comecei por refletir em métodos que inovassem e simplificassem a combinação de 3D com 2D para artistas, animador com pouca experiencia, tais como:
-Substituir o processo complexo de animação das expressões faciais 3D por simplesmente desenhá -las em 2D no modelo 3D.
-Transformar imagens 2D em 3D, embora já haja aplicações deste método, cujos resultados estão restritos a objetos simples e básicos.
- O movimento dos modelos 3D serem baseados e criados a partir da animação de rigs 2D.
-Animar os modelos 3D *frame* por *frame*, de modo a não ser necessário criar rigs complexos
-Aplicar o processo 2D de animação de *frame* por *frame* na animação 3D, sendo desnecessário a criação de rigs complexos. Este processo apesar de tornar animação 3D mais acessível para animadores 2D, apresenta contudo o inconveniente de aumentar a quantidade de trabalho a realizar.

Atendendo que os métodos anteriormente referenciados requeriam recursos e tempo que não possuía e só eram apropriadas a certas situações especificas, optei por progredir com animação modular e aplicar técnicas 3D em recursos 2D, de modo a ganharem profundidade e volume.

Com o objetivo de perceber qual seria o tipo de câmera mais indicado para observar claramente as animações sem denunciar os elementos do jogo como objetos 2D, pesquisei, e em resultado desse estudo, decidi fazer do jogo um Beat 'em up com a típica câmera *Side-Scroller,* garantindo diversas animações que podem ser visualizadas facilmente. Concluí que os jogos *turn-based*, necessitam inteligência artificial complexa, enquanto o Unity oferecia uma componente de inteligência artificial de movimento compatível com o estilo de *Beat 'em up*.

A partir de Homeworld Universe Mod Tools tentei criar um *Skydome,* mas decidi não o utilizar porque estava a ter dificuldades em implementá-lo no Unity e porque só se justificava a sua utilização se o jogador tiver completo controlo da câmera.

Experimentei criar um *rig* em Blender com um modelo grátis e criei e pus em prática um cenário MultiPlane Camera effect em papel, e percebi o quão complexo são estes métodos e quanto tempo requerem.



Teste de rig em blender

5.2 Design visual

5.2.1. Design das personagens

A cultura Celta inspirou o vestuário das personagens. E o seu design foi condicionado pelos meus limites de ilustração.

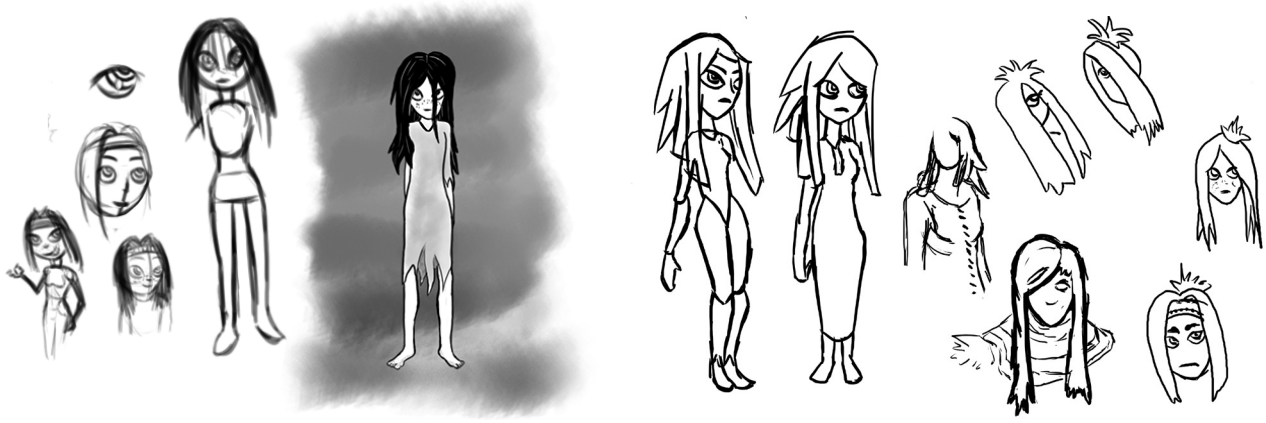
Para evitar gastar memória em desenhos da personagem a olhar para diferentes direções, o jogo faz flip à personagem, ou seja, roda a personagem 180º em relação a um eixo vertical o que implica que o seu design seja simétrico.

A Witch é uma rapariga jovem (14 anos) que sempre viveu na floresta, tendo pouco contacto social com a civilização, como tal o seu design é prático, inclui adereços da natureza e de bruxaria, um toque feminino dado pela utilização da cor de cereja e penas de corvo e um ar mais selvagem por estar descalça.

O design do bandido tem como objetivo dar-lhe uma aparência mais agressiva, o que é conseguido por ter o rosto obscurecido, calças com proteções de metal, troféus de caça no vestuário e aparência mais robusta.
O propósito dos esboços do bandido foi para definir o balanço entre um design rico de personalidade e uma estética genérica. Foi com a ilustração e animação do protótipo que apercebi-me, que se deve evitar desenhar a linha de contorno (*outline*), pois ao longo da animação esta é quebrada nas zonas de articulação da marioneta.

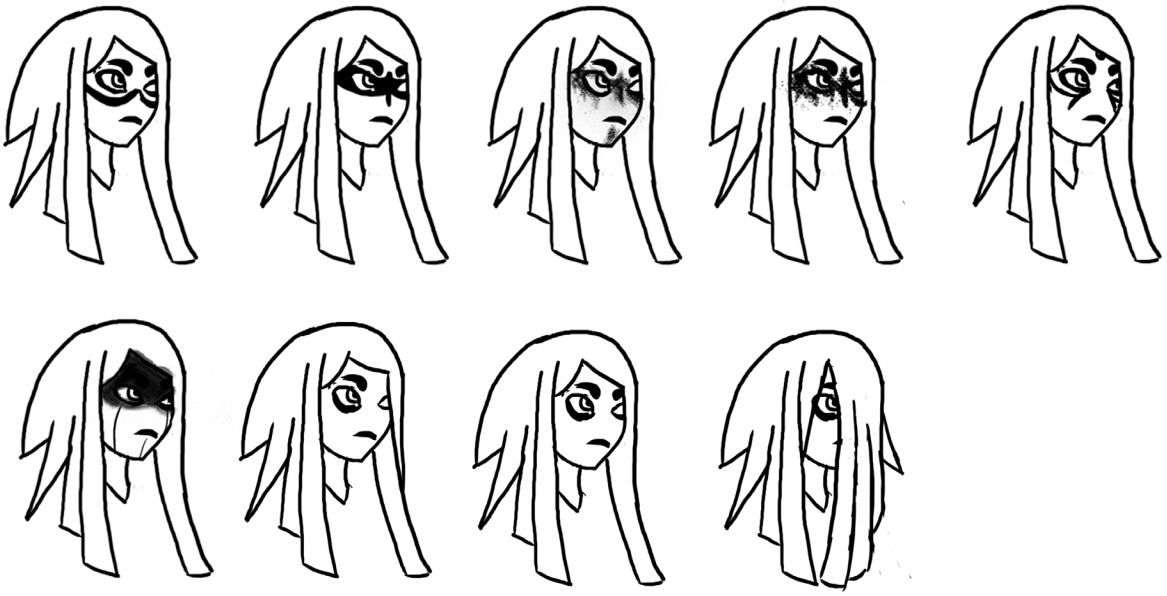
O design do bandido tem como objetivo dar-lhe uma aparência mais agressiva, o que é conseguido por ter o ros-
to obscurecido, calças com proteções de metal, troféus de caça no vestuário e aparência mais robusta.

Desenhei vários esboços da Witch para criar um modelo com uma forte silhueta e determinar o estilo de desenho como base para o resto do Jogo, tendo em conta que fosse acessível a minha habilidade de ilustração concluído com um estilo simples e cartoon.

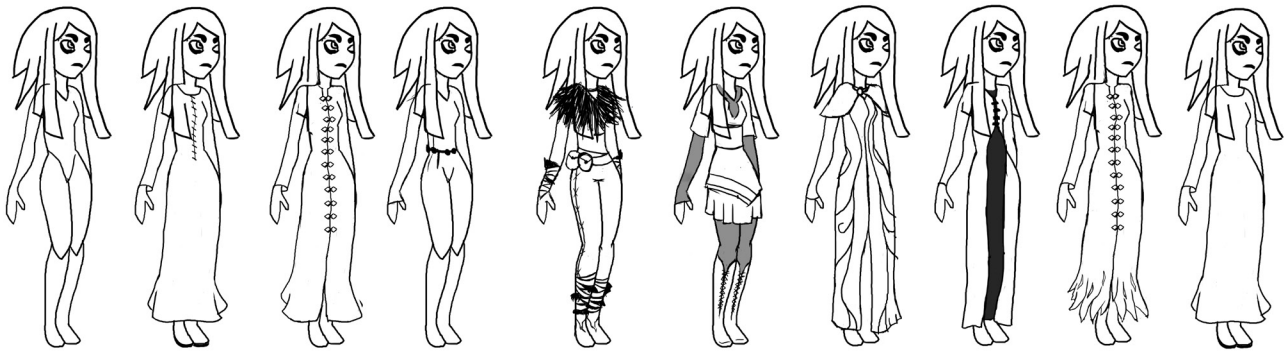


Esboços de Witch

Ilustrei vária cabeças e roupas com o âmbito de escolher as opções cujo design representa melhor o conceito da personagem, e tendo também em conta à *feedback*, selecionei a primeira cabeça e os dois design de vestuário ilustrados em testes cromáticos. Mais adiante é que exclui o vestuário de saia porque apesar de representar bem o elemento místico e simplicidade, iria provar como um desafio em animar a saias. Principalmente tendo em consideração que possuo pouca experiencia em desenhar e animar roupas.



Cabeças da Witch

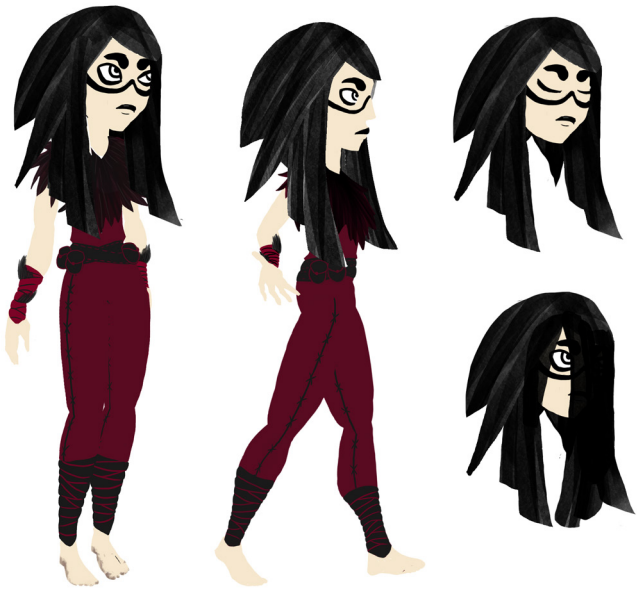


Vestuário da Witch

Realizei os testes cromáticos para escolher o padrão básico de cores e a sua distribuição no vestuário da Witch, de modo a reforçar o seu conceito e criar contraste com o cenário. Portanto escolhi vermelho cereja e preto em que cada uma representa respectivamente o elemento feminino e bruxaria
A partir dos testes cromáticos, decidi escolher as cores de base da Witch, de vermelho cereja e preto em que cada uma representa respectivamente o elemento feminino e *witchcraft*.

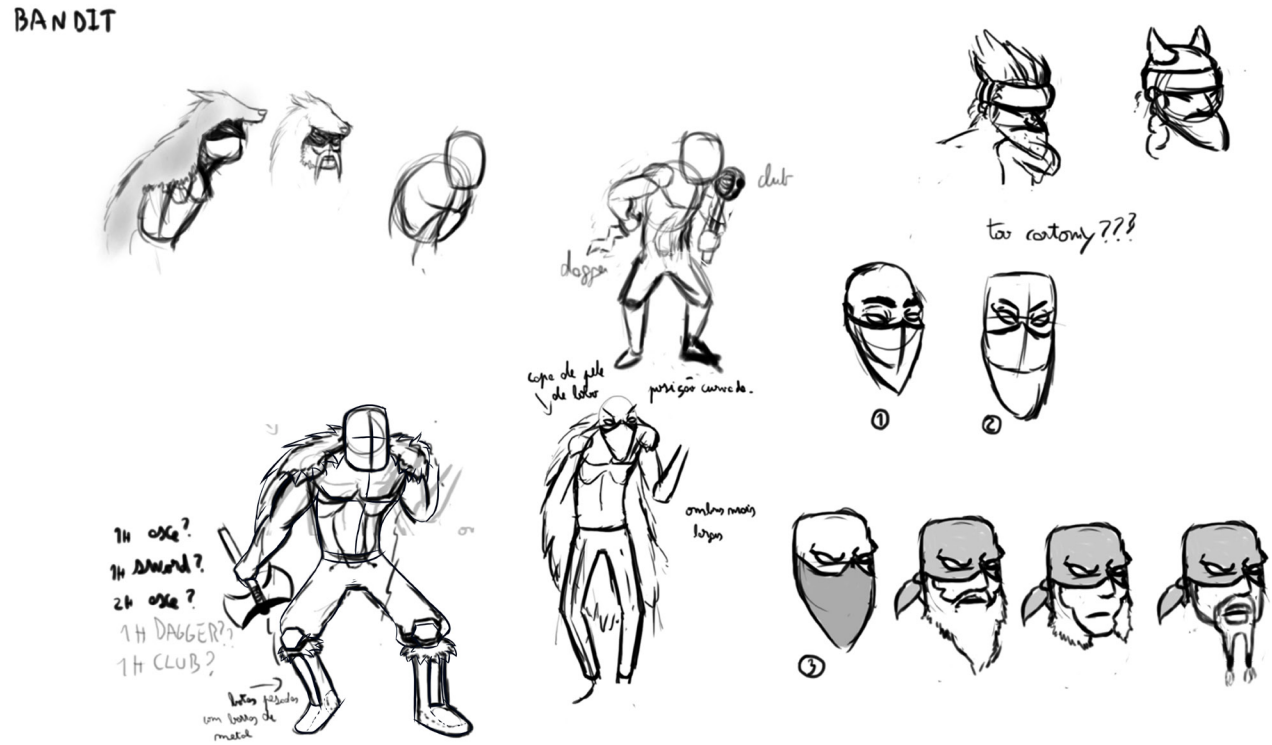
Teste cromático da Witch

Em conclusão, este processo resultou do Design final da Witch, em que reaproveitou-se o último design da cabeça para representar a Witch magoada.



Design final da Witch

Ao desenhar as personagens apercebi-me, que se deve evitar desenhar a linha de contorno, pois ao longo da ani-
mação esta é quebrada nas zonas de articulação da marioneta



Esboços do bandido

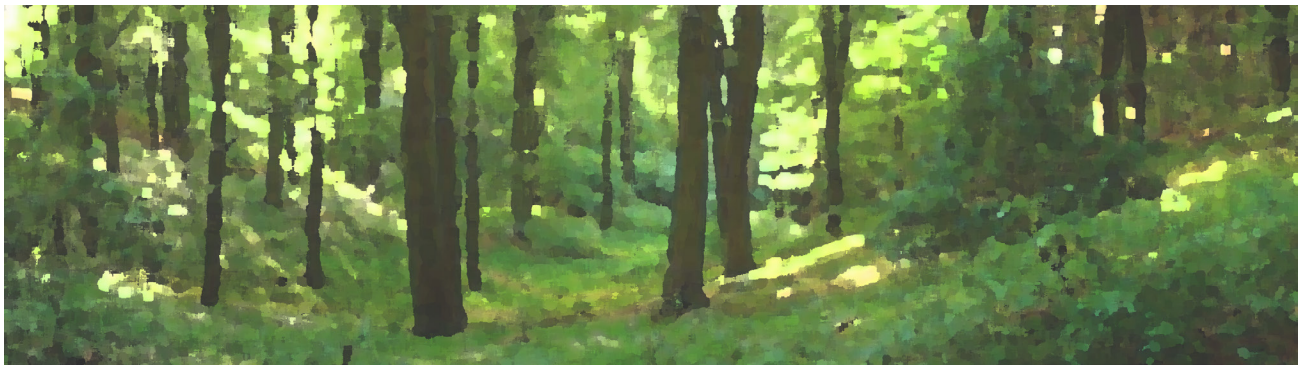
Protótipo do bandido

Design final do bandido

5.2.2. Design do Ambiente

Face à minha inexperiência na construção de cenários, e considerando que o local onde se desenrola o nível de jogo, decidi construir o cenário a partir de uma fotografia de uma floresta. Apliquei o filtro Pallete knife do Photoshop, para o cenário ficar mais consistente com as personagens e, por facilitar a edição não criando inconsistência gráficas.
Para aplicar o MultiPlane Camera effect, identifiquei os diferentes planos de profundidade da imagem e editei-as para criar os planos *foreground, middle ground* e *background*.

Foto do cenário sem filtro



Protótipo do *Background*

Protótipo do *Middle ground*

Protótipo do *foreground*

Assim que implementei o MultiPlane Camera effect apercebi que apesar de ter implementado as *outline* e *inline* do *foreground* e *middle ground* *para* oferecer forma, corpo, volume aos elementos ilustrados, estas quebravam a ilusão de profundidade e criava inconsistência visual entre os planos, e, cada plano precisava de ajustes para reforçar a ilusão de profundidade, tais como.

Para que o protótipo do *background* ocupasse completamente o fundo do jogo, tive que aumentar o seu tamanho, consequentemente criava árvores gigantes que quebravam a perspectiva e ilusão de profundidade. Portanto a partir da edição da imagem, tornei-a mais extensa.

Imagem do *Background* do cenário

Tanto como no Middle ground como no foreground, retirei conteúdo de modo a ser possivel visualizar mais claramente os diferentes planos.

Imagem do *Middle ground* do cenário

Tendo em conta que o *foreground* é o plano mais próximo ao jogador, decidi aumentar o diâmetro das árvores e decidi fazer cortes em forma de relva na base do foreground, com o propósito a fundir o *Foreground* com o terreno, ocultando que o foreground é tecnicamente uma imagem 2D assente perpendicularmente no terreno.

Imagem do *Foreground do cenário*

Criei um plano adicional que se localiza entre o jogador e as personagens, com o proposito de o imergir no jogo, pois esta cria a ilusão de que o jogador encontra-se a observar a ação no bosque.

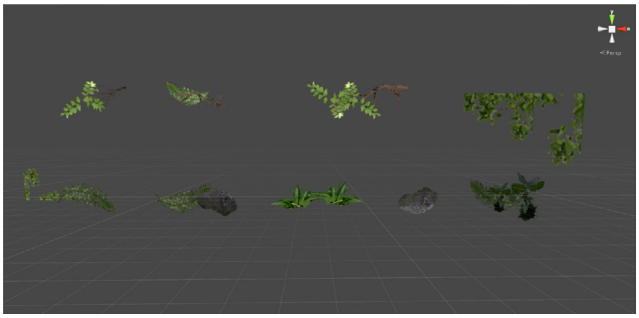


Imagem do plano de imersão

Imagem da visualização do plano de imersão através da câmera.

Construí a textura do terreno do cenário no jogo através da combinação de três texturas, tendo em consideração que o terreno teria que fundir com o *foreground*.

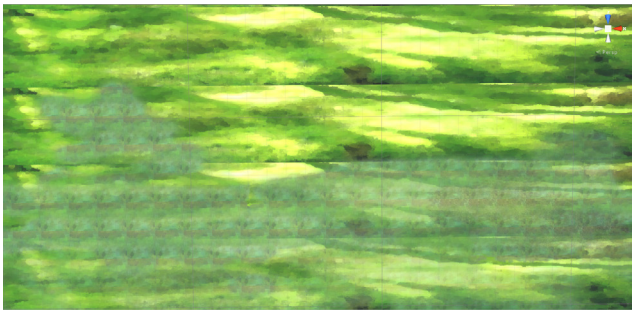


Imagem do terreno

Imagem das texturas usadas no terreno

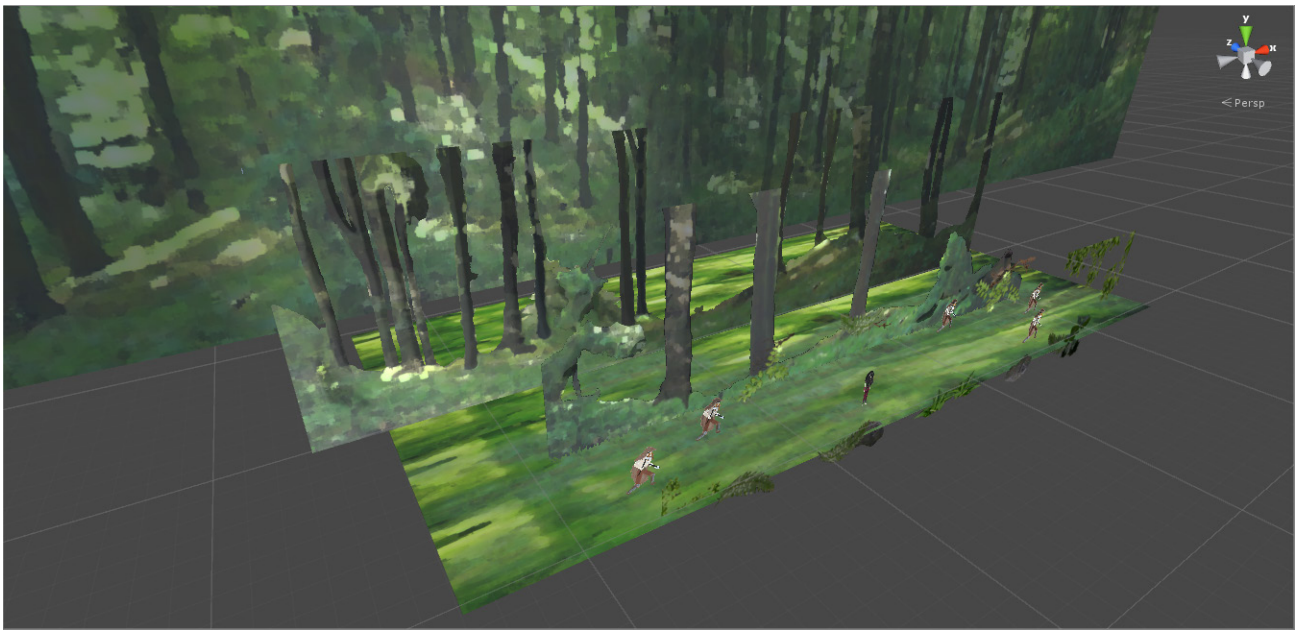


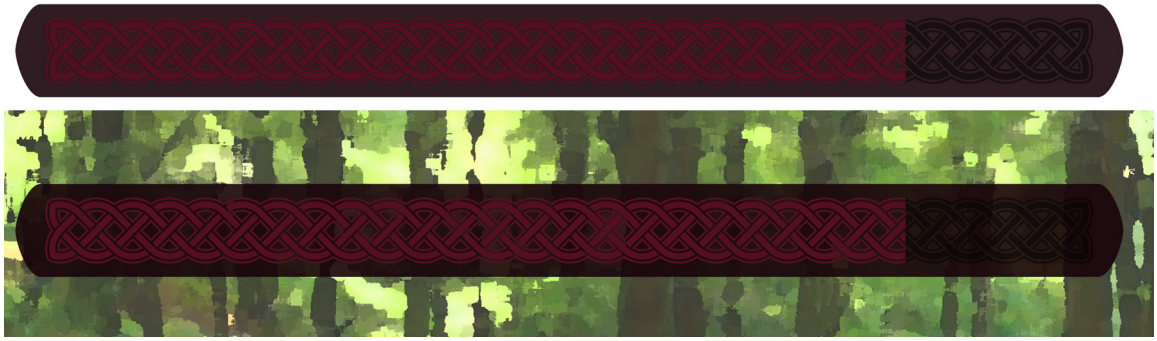
Imagem do cenário completo

5.2.3 . Design de GUI (Graphical user interface)

O design da barra de vida da Witch foi inspirado pelos *celtic ornaments*, em que a barra de vida é uma celtic *knotwork*, acompanhado com uma base meia translúcida para reforçar a distinção entre a barra de vida e o cenário. Para facilitar o reconhecimento ao jogador da ligação da Witch com a barra de vida, estes dois partilham o mesmo padrão de cores - preto e vermelho cereja.
Experimentei com vários protótipos, com o propósito de encontrar o design da barra de vida, cuja leitura fosse clara e fácil ao jogador, que condize-se com o visual do jogo e também oferece-se contraste com o cenário.

Protótipos da barra de vida

Componentes da barra de vida



Design final da barra de vida

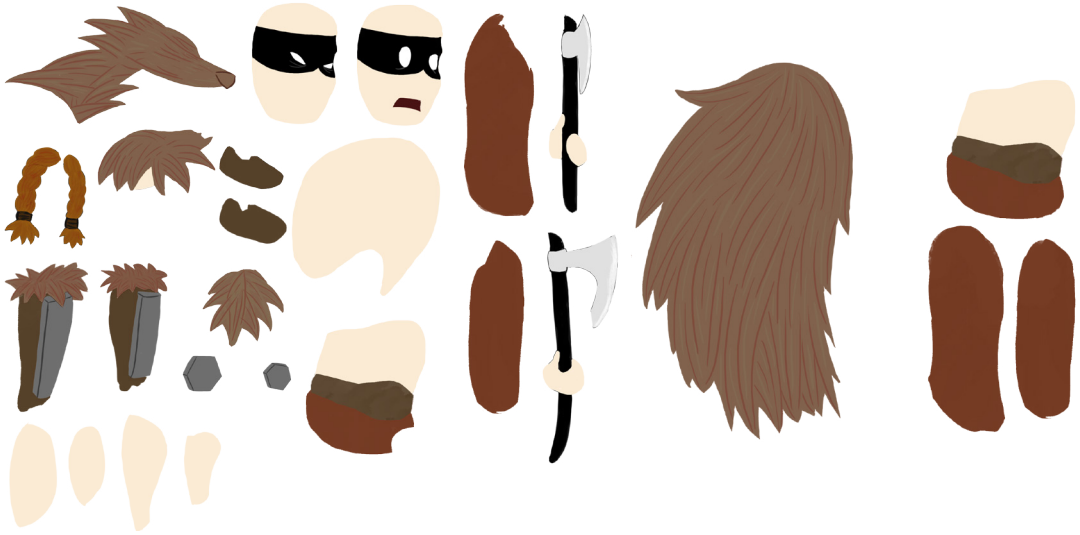
5.3. Produção das personagens

Explorei o programa Spriter e construí e animei um modelo, marioneta teste. Mais tarde, ao aperceber-me que a aplicação responsável para exportar a marioneta do Spriter para Unity foi criada recentemente por fãs e ainda tinha *bugs* e restrições de controlo da marioneta em Unity, optei por construí-la no *engine* 3D do Unity. Assim, não só os *rigs* podem ser 3D, como também tenho controlo total na marioneta. Por exemplo, através de código ou das ferramentas do Unity, consigo criar eventos do jogo que alteram partes da marioneta. Em contrapartida o rig da marioneta é básico, portanto não os consigo construir com uma quantidade grande de diversos elementos sem exercer muita carga de trabalho ao processador.

As personagens são desenhadas e divididas em partes da marioneta no Photoshop, que mais tarde são guardadas em formato de *texture map*. Através do Unity estas são divididas, em que todos os seus pedaços são categorizados e atribuídos ao rig marioneta, formando a personagem. A desvantagem deste processo, é que só é possível determinar se os desenhos da personagem estão bem construídos para serem animados no final deste processo. O que significa que o menor erro detetado implica corrigir e repetir todo o processo.

Para perceber como profissionais constroem os *rigs* e ilustram as personagens em animação modular, adquiri os vários segmentos que compõe as personagens do Dragon Crown. Através de uma imagem da personagem Amazon e Fighter como base, e, com os segmentos colecionados, reconstruí as personagens de modo a ajudar-me a compreender a lógica e cuidados que devo ter na criação de animação modular.

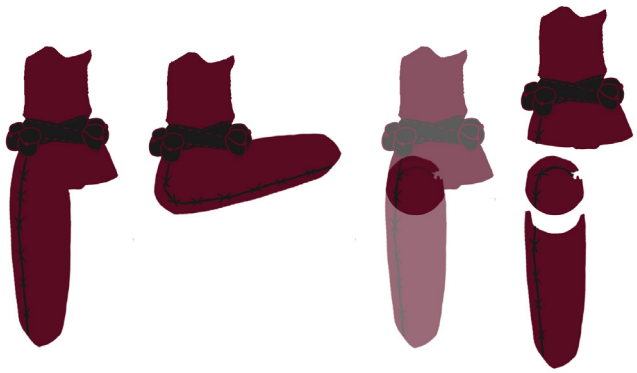
Texture maps da Witch



Texture maps do bandido

Uma das desvantagens de usar rig marioneta, é que a ilustração das suas zonas de articulação necessitam de cuidado e atenção. Por exemplo, na linha de costura que atravessa as calças da Witch, não importa o ângulo entre as pernas, as linhas de costura têm de conectar-se entre si. Mas no caso da articulação entre a anca e a perna da Witch, quando esta se levanta, supostamente o tecido e a linha de costura deviam ficar deformados, sendo necessário acrescentar uma peça específica ao rig para representar a deformidade.

Para indicar ao jogador que a Witch tem pouca vida, 30% da barra de vida ou menos, a personagem está programada para ficar com cabelo à frente da cara. Tipicamente isto significa ter que criar um conjunto repetido de animações da Witch com o cabelo à frente da cara, principalmente em animação de sprites, mas como estamos a utilizar animação modular, só é necessário acrescentar as peças de cabelo ao rig, poupando muita memória.



Exemplo da deformação

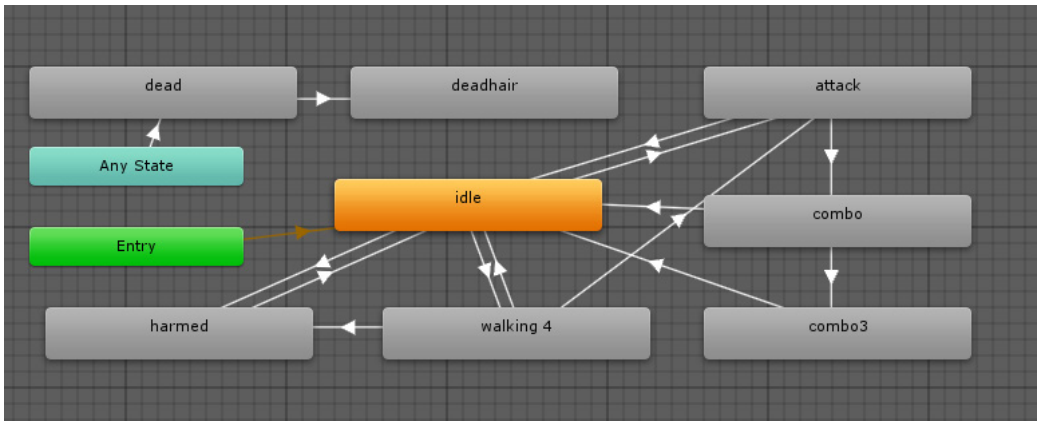
Witch com pouca vida

Para facilitar o processo de animação, tentei incorporar código no jogo para poder utilizar *rigs* IK (Inverse Kinematics), contudo, depois de alguns testes, os rigs IK tiverem que ser descartados porque só funcionariam com qualidade em animações simples.

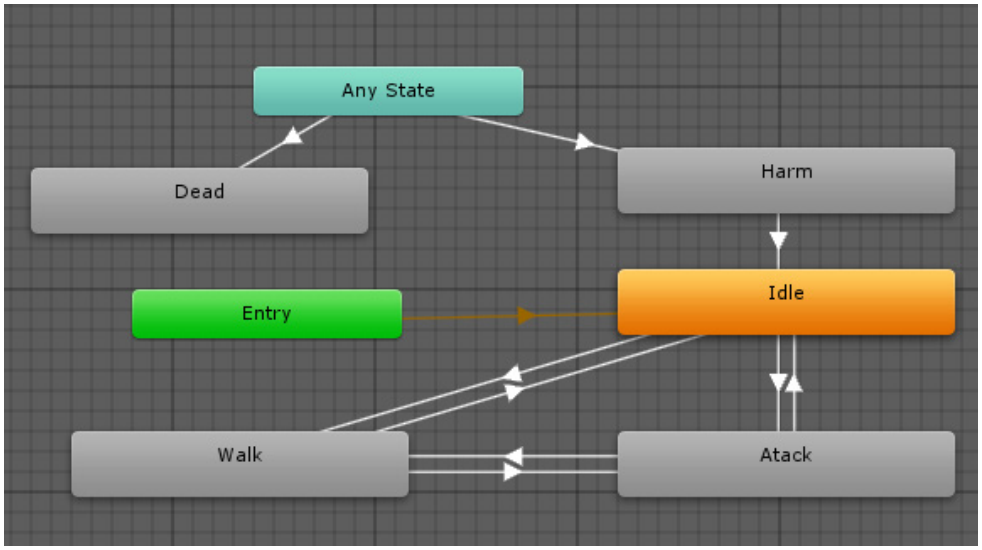
Animação das personagens

Tipicamente jogos preferem ter entre 30 a 60 fps, como tal decidi criar as animações no jogo a 60 fps, sendo o ideal para jogos de computador. Estas animações resultam da combinação de animação modular e animação de sprites, o que possibilita que a da troca sequencial dos fragmento da marioneta, não só tornando o seu movimento mais fluido, mas também cria a ilusão de movimento fluido de sprites 2D no terceiro eixo de coordenadas (3D)

Em jogos, as animações estão naturalmente condicionadas às ações do jogador, portanto o jogo tem que conter mapas que indicam em que condições as animações são iniciadas e quando é que estas podem



mapa de animação da Witch



mapa de animação do bandido

dead- morrer
deadhair- animação de game over
idle- respirar, enquanto espera por ordens
harmed, harm- personagem a ser magoada
walking, walk- andar
attack, atack- atacar
combo attack- ataque consecutivo

Os mapas de animação entre a Witch e o bandido são diferentes, visto que ao contrário do bandido que pode ser harmed em qualquer momento, a Witch só pode ser *harmed* quando não está a atacar. Sem esta diferença o jogo ficaria mais difícil.

As personagens só andam na direção este e oeste, porque cheguei à conclusão que precisaria no mínimo do triplo do trabalho para as desenhar a andar para o norte e sul, criar os seus rigs e as suas animações, pelo que não teria tempo. Não implementei a animação de *dash*, porque no final desta animação a Witch ficava sempre na posição errada e não consegui corrigir este problema. Apesar de ter escrito um código que corrigia a posição da Witch, havia algum elemento desconhecido no Unity que alterava a posição.

Não criei as animações de magia em evocar criaturas, de projéteis, laser, porque requeria adição e modificação do código do jogo, além de diminuir o desafio do jogo retirando-lhe interesse.

Não implementei as animações de salto porque a Inteligência artificial responsável pelo movimento dos bandidos é uma componente do Unity designada por Nav Mesh Agent, mas esta não inclui a habilidade de os fazer saltar.

Animações da Witch

walking

harmed

dead

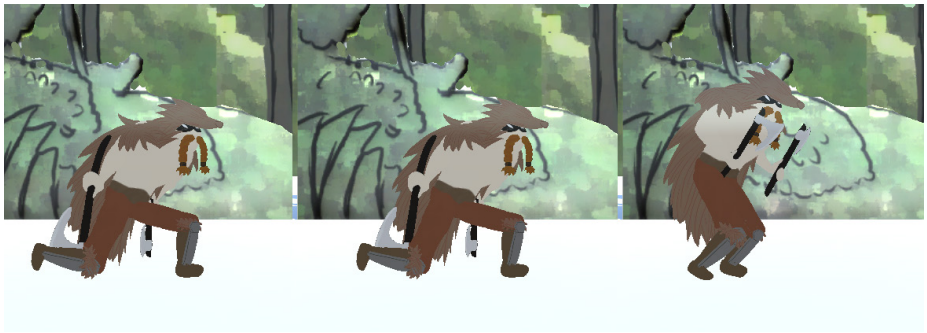
deadhair

attack

combo

combo 3

Animações do bandido



attack

walk

harmed

dead

No jogo, a luz do sol é representado por uma *Directional light*, e os efeitos especiais do golpe da Witch, chamas, possuí uma *Point Light*.

Com a intenção de reforçar a ilusão objetos 2D terem volume, decidi aplicar material ao cenário, personagens, assim estes são afetados pelas fontes de iluminação no Unity.

A desvantagem da aplicação de um material numa personagem, estamos ao mesmo tempo indicar ao unity que só deve renderizar a face da frente da personagem. Portanto assim que a personagem muda de direção, faz flip, a face de trás da personagem fica revelada mas não é renderizada, ou seja não é visível, fica transparente. Para que a ambas faces sejam renderizadas, tive que alterar o código do *shader* do Unity e aplicá-lo às personagens.

No âmbito de introduzir a técnica de iluminação 3D em objectos 2D, de modo a criar a ilusão de volume em imagens, Desenhai e apliquei os *normal maps* no cenário e das personagens. Contudo quando a personagem faz flip, a direção das *normal lines,* responsáveis por calcular como a luz afeta a superfície da imagem, ficam no sentido oposto, portanto induz uma reação errada à iluminação. Apesar deste dilema ter várias soluções, não as consigo por em prática. Tal como:
-Customizar o *shader* de modo a corrigir a direção as *normal lines,* mas requer conhecimento que não possuo.
-Criar atrás na personagem uma cópia com as *normal lines* na direção correcta, mas deste caso é uma solução ineficiente, porque exerceria muita carga de trabalho ao processador.
-Através num software 3D, tornar os fragmentos das personagens como *double layered objects,* ou seja, em imagens em que ambas superfícies possuem *normal lines.* Apesar desta ser melhor a solução, requeria que repetisse o processo de criação e animação das personagens, a qual que não tenho tempo.

O *background* não tem um normal map, para evitar com que este seja afetado pelas iluminação do golpe da Witch, visto que este supostamente encontra-se longe das personagens.

Imagem do normal map da Witch

5.5. Câmera do Jogo

Implementei uma câmera Side-Scroller, que acompanha a Witch no eixo de ordenadas X. A câmera encontra-se próxima à Witch, permitindo ao jogador observá-la bem. Mas, com a presença de inimigos, o jogador necessita de um maior campo de visão para obter informação do espaço e o posicionamento dos bandidos. Portanto, sempre que um destes aparece na visão da câmera, esta afasta-se gradualmente do cenário aumentando o seu campo de visão.

Tecnicamente a câmera é constituída por duas câmeras, uma que é responsável por renderizar o GUI (Graphical user interface) e outra que renderiza o restante. Este método do Unity é de modo a garantir que não importa o que haja entre a câmera e GUI, este aparece sempre sobreposto ao restante .
O modo como a câmera é utilizada durante o jogo tem um impacto sobre a perceção do jogador em relação às animações, tal como reforçar o peso e o valor das animações. Considerei fazer com que câmera rodasse ligeiramente ou que se aproximasse às personagens em determinadas animações, para reforçar o seu peso e torná-las mais dinâmicas, mas conclui que acabaria por distrair e atrapalhar o jogador.

Para reforçar a ilusão de volume das personagens e cenário 2D, criei *oclusion maps* e *normal maps*. Portanto desenhei e apliquei *occlusion maps* nos materiais, o que fez com que as personagens ganhassem sombreamento. Realizei testes com *normal maps*, mas apercebi-me que não tinha a experiência para as criar com boa qualidade para que o resultado fosse realista.

5.6. Progamação.

Uma das razões para ter escolhido Unity era porque funcionava com código javascript, entretanto preferi aprender e utilizar antes o código C#, por me aperceber que havia um maior apoio neste código da comunidade Unity, com diversos conselhos e tutoriais úteis para o desenvolvimento do jogo.

Ainda implementei um sistema de combos, com o propósito de mostrar uma maior galeria de animações e tornar o jogo mais interessante. Todavia este sistema requer do jogador muita coordenação para efetuar combos.
Na câmera tentei implementar um código que por cada bandido que entrasse na sua visão, esta afastava-se para aumentar o campo de visão. Mas isso envolvia descobrir constantemente as novas coordenadas dos bandidos, o que sobrecarregava o processador, tornando o jogo lento.

A introdução de um sistema de alteração do mundo, corrupção da natureza, maldições no jogo segundo o uso excessivo dos poderes da Witch, tornaria este nível de jogo mais interessante e divertido, contudo tal processo não se mostrou exequível dado exigir demasiado tempo para criar o seu design e código.

Os códigos que implementei e adaptei foram baseados nos tutoriais dos canais de youtube Casanis-Plays e C Sharp Accent Tutorials

5.7. Audio

O áudio ajuda a imergir os jogadores no jogo, portanto também ajuda imergir, sentir a animação. Como tal escolhi os seguintes sons com a licença creative commons no site https://www.freesound.org/.

Music:
-Uma música céltica, que fosse feita de modo a que pudesse ser repetida sem que a sua melodia sofre-se disrupções.
Fantasy theme de Slaking-97

-Música que denucia a morte da Witch .
impact-lost-in-the_woods de Death Alexnekita

Ambience:
-Sons da floresta
Forest Sounds de Porphyr

Foley:
Importante para animação, porque reforça a percepção dos jogadores no que respeita às ações das personagens e ajuda a fazer sentido os movimentos instantâneos.

-Witch magoada
dying female de 11linda

-Ataque do Bandido
Whooshes Whips and Swooshes de Tiger_v15

-Andar da Witch
walk-forest de Jankoehl

Não implementei os sons dos bandidos a andarem, porque criaria uma superabundância de sons.

Sound Effects (SFX)
-Magia do fogo
Basic Fire whoosh 3 de Niedec

-Som de uma trompa, ocorre para denunciar o aparecimento, ataque dos bandidos.
battle horn de Porphyr

6. -Conclusão

O objetivo desta dissertação era combinar as técnicas simplificadas e refinadas 2D com as técnicas dinâmicas 3D, de modo a superar a estética plana, e em criar novos sabores na animação no contexto de jogos. O que conduziu ao longo da pesquisa de animação 3D e 2D e outros jogos à coleção de técnicas relevantes ao tema da dissertação. Criou-se um nível do jogo Lil´Witch através da *selecção* e aplicação das técnicas animação modular, iluminação 3D em 2D e MultiPlane Camera effect. A partir deste estudo, conclui que, para criar resultados com excelente qualidade é necessário experiência, mas assim que uma pessoa domina as técnicas pode revelar novos potenciais em jogos, e, principalmente em animação. Estas técnicas permitem modificar o processo de animação criando alternativas o que o torna mais adaptável às necessidades e qualidades dos animadores e dos projetos, e, em consequência, possibilita a aplicação de técnicas que eram antigamente incompatíveis ou ineficientes. Por exemplo, em animação de *sprites*, seria necessário criar uma *normal map* ou *oclusion map* diferente para cada sprite, ao contrário da animação modular.
É ainda de referir que as técnicas 2D com 3D, não só permitem enriquecer animação, mas como também a estética do jogo e usar novas mecânicas de gameplay pelo que deve merecer mais atenção.
Portanto tendo em conta que consegui implementar as técnicas propostas, estou extremamente satisfeito com o que aprendi sobre animação, campo de estudos 3D e Design de jogos. Tal como fiquei satisfeito com os resultados do prototipo de jogo, apesar de reconhecer que a criação e implementação dos *normal maps* na técnica de iluminação 3D em 2D, necessitavam de ser refinadas, sendo mais lamentoso tendo em consideração que tinha conseguido detectar o problema e a solução mas não tive tempo de a implementar.

6.1 -Dificuldades

Durante o processo da dissertação deparei-me me com alguns problemas que dificultaram o seu progresso, nomeadamente a falta de experiência e não ter um conhecimento sólido na área de animação 3D e de pintura digital. Foi necessário aprender a escrever código C# e o código de Unity, que constitui o maior desafio, pois exigiu a título exemplificativo, aprender como este comunica com a estrutura do editor do Unity e o comportamento das suas várias funções e classes. As dificuldades anteriormente elencadas e as inerentes à construção do jogo, das quais destaco a dificuldade em me concentrar numa só tarefa/área por existir uma ligação natural entre as diversas áreas do jogo, tais como design, desenho, *rig*, animação e principalmente programação, em que o desenvolvimento de uma condicionava o processo das outras. Assim, os progressos feitos não se traduziam em resultados imediatos e foi necessário aprender a trabalhar de uma forma atenta e globalizante o que se revelou ser um desafio para cumprir com a calendarização. Face ao exposto, apenas foi possível implementar os testes de usabilidade quando a estética e a animação do jogo estava completa, pois tendo em conta o âmbito desta dissertação, seria irrelevante aplicá-los com um protótipo de baixa fidelidade.

6.2 -Alterações no futuro.

Forem colecionadas várias técnicas para criar um nível num jogo de 2D com 3D, contudo este ainda têm espaço para melhorias. Tais como:

-Em ordem a tornar o jogo mais interessante, teria que ser implementado o sistema de runas, corrupção da natureza e um melhor sistema de combate.

-Melhorar o *rig* de animação modular, de modo a conseguir suportar animações mais complexas sem adicionar muita carga de trabalho ao processador, e, conseguir usufruir todas as ferramentas e funções do Unity.

Glossário

gameplay - É a interação entre os jogadores e o jogo, segundo as regras do jogo.
test users - São usuários que testem um produto, cuja experiencia é recolhida e analisada.
feedback - É uma resposta, consequência numa ação do usuário.
modelos proxy - Modelos de baixa resolução que representa os modelos finais.
software - É o termo geral para os diversos programas usados para operar o computador.
hardware- São as componentes físicas, eletrônicas que constituem o computador.
reentrancias - Angulo ou curva para dentro, por exemplo uma cavidade.
mesh - Coleção de vértices, arestas e faces, que definem a forma num modelo em computação 3D.
Key frame - É uma frame que indica na linha de tempo o inicio ou final de um movimento, transição
Key pose- frames que contem as poses das personagens, que definem a ação da personagem.
Processador - É uma componente de hardware do computador, responsável pelos cálculos, processamento de dados e execução de tarefas. Portanto a velocidade de processamento do computador depende do seu processador.
Placa gráfica - É a componente de hardware do computador responsável pela construção e apresentação de elementos gráficos, por exemplo a da imagem do ecrã.
Disco rigido - É uma componente de hardware do computador, responsável pelo armazenamento de dados, ou seja a memória do computador.
Bugs - É um erro no funcionamento comum do software ou hardware do computador.
skydome- metodo para simular o fundo de paisagens, por exemplo a imagem de edificios, montanhas , ceu são projectadas do interior num hemisfério ou esfera, portanto com a camera do jogo dentro do skydome, cria a ilusão do fundo de paisagem estar a uma grande distancia.
npc- non playable chracter, personagens que não são controladas pelo jogador.
modelos proxy- modelo criado para representar o original.
plugin- um componente de software que adiciona uma função à um programa preexistente.
npc- non playable chracter personagens que não são controladas pelo jogador

Lista de imagens

Esta Lista contém os links das imagens .

Figura 1- teste básico de animação
https://indiaholdenap.wordpress.com/2014/09/30/bouncing-balls/

Figura 2 - Exemplo de Pose to Pose
https://shuyuq.files.wordpress.com/2013/11/untitled-1.jpg

Figura 3- Exemplo de storyboard
http://andothernonesuch.blogspot.pt/2012/08/adventure-time-storyboard-test.html

Figura 4- Pipeline 3D
Beane A. (2012). 3D animation essentials

Figura 5- Exemplo de Polygon mesh constituido por Quad
http://www.3dtotal.com/tutorial/1868-3ds-max-character-creation-chapter-1-photoshop-v-ray-by-andrew-hickinbottom-female-pin-up-olivia?page=3#.VFEJa_nF_84

Figura 6- Gráfico do Uncanny valley
https://en.wikipedia.org/wiki/Uncanny_valley

Figura 7- Exemplo de UV map
http://lart3d.com/3dgallery/3d12mg.html

Figura 8- Exemplo de Normal map
https://en.wikipedia.org/wiki/Normal_mapping

Figura 9- Exemplo de um rig
http://www.ronald-fong.com/blog/portfolio-item/rigging-for-animation-demo-reel/

Figura 10 - Exemplo de Smear
http://toonamir.blogspot.pt/2011/08/influences-on-barley-way-more-on.html

Figura 11 - Exemplo com hitspot e overshoot
http://tralfaz.blogspot.com/2012/12/hare-do-smears.html

Figura 12 - Exemplo de Motion-Capture Animation
https://www.behance.net/gallery/16734909/Motion-Capture

Figura 13- Exemplo de Raytracing
https://blog.codinghorror.com/real-time-raytracing/

Bibliografia

Chopine A. (2011). 3D art essentials. Elsevier, Focal Press
Beane A. (2012). 3D animation essentials
Thomas F. & Johnston O. (1981) The illusion of life. Italy, Walt Disney Productions

Webgrafia

Multiplane Educator, guide recuperado em http://www.waltdisney.com/sites/default/files/Multiplane-GuideCurriculumPacket_Final.pdf

https://simonschreibt.de/game-art-tricks/

https://www.youtube.com/channel/UC0JB7TSe49lg56u6qH8y_MQ

http://wiki.polycount.com/wiki/Polycount

https://www.youtube.com/watch?t=1&v=y-chi097uV4

http://polycount.com/discussion/121144/convincing-3d-that-looks-like-2d-wow/p8

http://chrisoatley.com/making-of-paperman/
http://www.fastcompany.com/3006276/open-company/trying-woo-animators-disney-accidentally-invents-paperman-method

http://www.gamasutra.com/blogs/HermanTulleken/20150729/249761/Color_in_Games.php

http://www.gamasutra.com/view/feature/132486/king_of_2d_vanillawares_george_.php?print=1

http://www.gamasutra.com/blogs/PeymanMassoudi/20150811/250937/The_Challenge_of_Responsiveness_VS_Naturalness_in_Game_Animation.php

http://www.stateofplaygames.com/work/lumino-city/

http://stoicstudio.com/animation-process/