# Design de Jogos Digitais

## Aula 09 - Prototipação e Testes

# Apresentação

Alô, amigos da rede jogo! E chegamos a mais uma aula! (Melhor tentar de novo. Ficou com cara de jogo de futebol!)

Bem-vindos a mais um momento de aprendizado! (Muito formal! Não é bem a minha cara.)

Olá, meu povo! (É o que temos para hoje!)

Hoje nossa aula está focada em tarefas essenciais no desenvolvimento de um jogo: prototipação e testes! Apesar de serem atividades fortemente interligadas, elas têm suas particularidades e geralmente são desenvolvidas por equipes diferentes. A prototipação é associada à construção de produtos, enquanto os testes têm um caráter de validação, para verificarmos se partes do nosso jogo contêm erro, se fazem o que determinamos, dentre outras possibilidades.

Essas atividades iniciam desde a fase conceitual, principalmente a prototipação, e perduram ao longo de todo o projeto, com os testes intensificando mais ao final do desenvolvimento. Mas vamos falar sobre isso com mais calma ao longo da aula! Sempre que possível, veremos as relações entre essas ações, a equipe de desenvolvimento e as fases de produção do jogo.

É hora da ~~prototip~~ação!

Objetivos

- Entender os conceitos de Prototipação e Testes e sua importância no processo de desenvolvimento
- Conhecer como funciona o processo de prototipação dentro do design de um jogo
- Diferenciar tipos de protótipos de jogos
- Conhecer os diferentes tipos de testes presentes na produção de um jogo

# 1 - Propósito dos testes e da prototipação

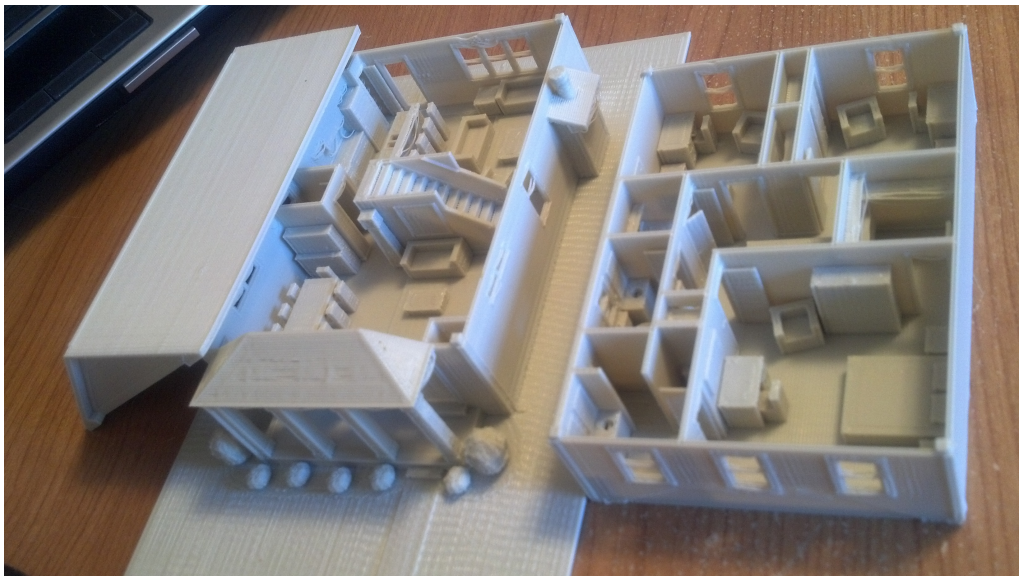
Que tal iniciarmos os nossos estudos tentando responder a três perguntas:

- O que são protótipos e testes?
- Para que servem?
- Como se relacionam com o processo de desenvolvimento de um jogo?

De acordo? Vamos nessa!

Primeiramente temos as definições. Um protótipo é um produto intermediário construído ao longo do desenvolvimento e que representa uma versão simplificada da ideia de trabalho ou ainda uma parte específica do projeto. Veja que esse não é um conceito exclusivo dos jogos: maquetes são protótipos da construção civil e da arquitetura, assim como modelos da galáxia feito com fios e isopor também são um protótipo de como o nosso universo é organizado. A ideia do protótipo não é ser uma versão final do produto, embora seja um passo importante no entendimento e na visualização de como a versão definitiva do projeto tomará forma.

**Figura 01** - Os protótipos são uma representação da ideia de trabalho e servem para facilitar a visualização de como a solução final será.

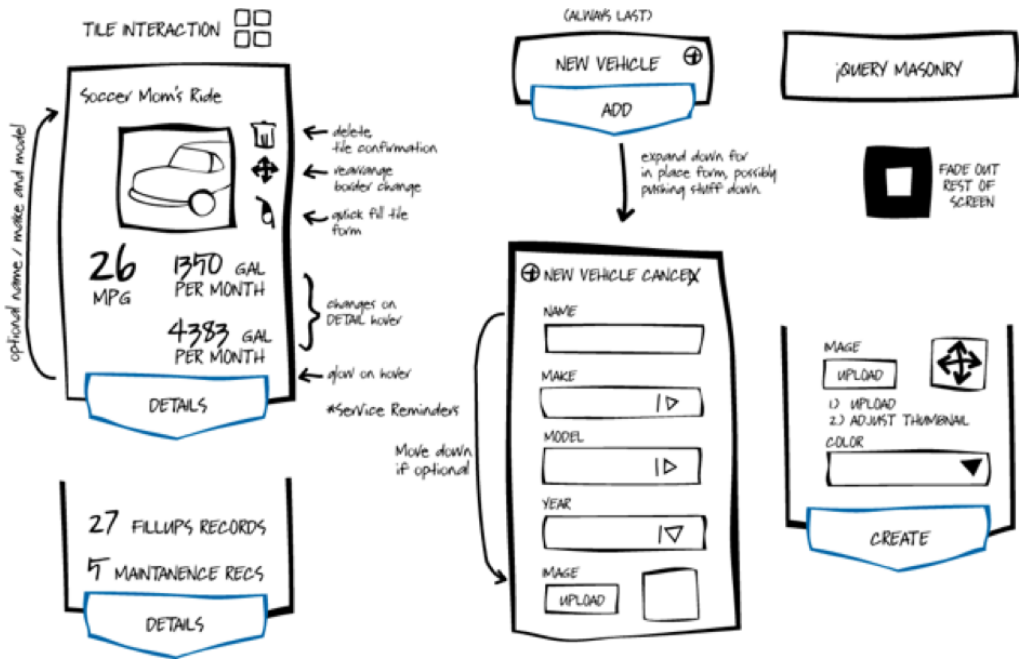


**Fonte**: http://www.dulaneydraftinganddesign.com/Quick_Prototypers.htm. Acesso em 28 out. 2015.

Os protótipos são construídos com um **objetivo específico**: podem ser um rascunho ou maquete utilizados em sessões de *brainstorm* para explicar conceitos ou ideias, ou ainda facilitar a visualização de uma mecânica de jogo cuja descrição textual ou falada não é tão intuitiva. Independentemente da sua finalidade, o protótipo possui um escopo limitado, não sendo uma implementação completa do produto, apenas o suficiente para cumprir o seu propósito.

Imagine que estamos fazendo um jogo de estratégia e desejamos que ele possua uma tela simples e com poucos controles na HUD para facilitar a visualização do jogador. Para avaliar se a tela está limpa (poucos itens expostos) ou fácil de manusear, não é preciso ter todo o jogo implementado! Um esboço dela feita em papel ou em um software de desenho, ou até mesmo uma versão implementada da interface de forma computacional, apenas para visualização, já pode ser o suficiente para apresentar uma proposta válida para discussão, e a partir daí evoluir a interface para o seu conceito final. Uma vantagem dessa abordagem é que, por não se tratar de um produto completo, o protótipo é bem mais barato e rápido de construir, o que permite manter os custos do desenvolvimento reduzidos em fases iniciais do projeto.

**Figura 02** - A prototipação de interfaces em papel é uma forma extremamente rápida e simples de apresentar uma ideia de visualização para o jogo.




**Fonte**: https://msdn.microsoft.com/en-us/library/hh404087.aspx. Acesso em 28 out. 2015.

O uso de protótipos também auxilia na **redução de riscos** de um projeto, principalmente no tocante a orçamento e cronograma. Imagine a seguinte situação: você está desenvolvendo um jogo de FPS futurista e deseja incluir uma mecânica de movimentação por teletransporte. Você desenvolve a ideia, pensa nas regras e nas formas como o jogador ativará essa mecânica. A partir daí você precisa testar se ela realmente funciona em um ambiente de jogo FPS, se o jogo permanece balanceado, se não existe uma estratégia dominante que apareça devido à inclusão dessa mecânica. Você pode até tentar intuir isso de sua experiência, mas a melhor forma de realmente perceber como a mecânica afeta a jogabilidade é através do próprio jogo!

Mas, e aí? Se o desenvolvimento fosse feito diretamente na estrutura do jogo, você teria que esperar toda a parte de recursos gráficos gerados pela equipe artística, os modelos 3D, o design de nível da fase e a implementação de todos os sistemas para testar o jogo. Parece muito tempo para começar a testar uma mecânica, não é mesmo? E se após tudo isso você descobre que a mecânica não funciona bem e precisa tirá-la do jogo? Todo o trabalho que foi dispendido na implementação dela (programação, criação dos efeitos visuais, sonoros, etc.) vai para o lixo. E isso significa dinheiro gasto (salário da equipe, custo de aquisição de material, etc.) e tempo da equipe, que poderia trabalhar em outras partes do projeto! Já consigo até ver o produtor dando um grito e batendo na mesa.

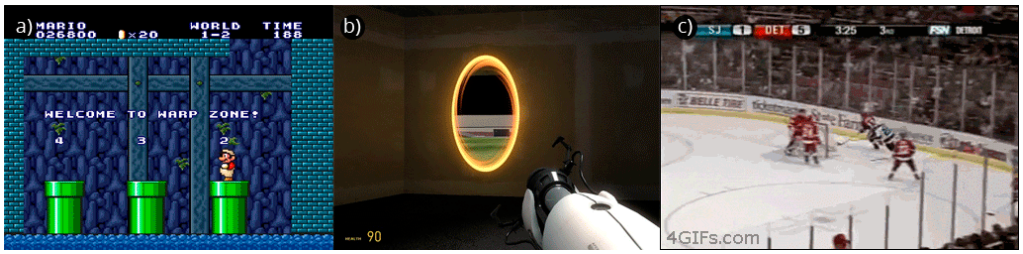
Para evitar esse tipo de situação, as equipes constroem versões simples da ideia que querem: um esboço de nível ou espaço de interação sem artefatos gráficos com os sistemas essenciais de movimentação do jogo já seriam suficientes para testar a mecânica. Afinal, a jogabilidade não depende da qualidade gráfica, não é mesmo? Dessa forma, rapidamente é possível construir um ambiente no qual a mecânica pode ser testada e validada, sem necessidade de investimento de toda a equipe do jogo na tarefa. No exemplo do teletransporte, poderíamos fazer um nível bem simples, até mesmo usando um modelo já pronto no motor do jogo, e fazer o teletransporte sem efeitos visuais, apenas reposicionando o personagem no nível. Após alguns testes, teríamos uma ideia de como isso funcionaria dentro do jogo, e a partir daí poderia ser tomada uma decisão com relação ao investimento no desenvolvimento dessa mecânica dentro do jogo ou não.

A prototipação está totalmente alinhada com a ideia de **iteração rápida e contínua** do processo de design, como vimos na primeira aula da disciplina. Como eles podem ser construídos e descartados rapidamente (devido ao baixo custo), é possível realizar vários ciclos de validação da ideia, levando a uma decisão mais rápida sobre o formato final da mecânica. Voltando para o exemplo: com o nível simples construído e a movimentação definida, nós poderíamos fazer um protótipo onde o teletransporte fosse:

1. a) acionado em um ponto fixo do mapa;
2. b) um item que o usuário coleta no nível e seja acionado como uma arma;
3. c) um comando natural do jogador (como pular ou abaixar) acionado a qualquer momento por uma tecla de comando.

A única mudança entre os três protótipos seria a forma de acionamento, e como não é necessário desenvolver nenhuma arte diferenciada entre eles, rapidamente as três ideias poderiam ser testadas. Inclusive elas podem ser implementadas de forma paralela por equipes diferentes, o que agilizaria a validação do formato mais adequado para o jogo.

**Figura 03** - Nosso teletransporte poderia ser feito de várias formas: pontos específicos na fase (a), através de um item ou arma (b) ou de forma natural (c)




**Fonte**: a) http://gaming.wikia.com/wiki/Warp_Zone. Acesso em 28 out. 2015.
b) http://www.moddb.com/mods/garrys-mod/addons/portal-gun-with-models. Acesso em 28 out. 2015.
c) https://media.giphy.com/media/10l0xtwCH8Ix9e/giphy.gif. Acesso em 28 out. 2015.

O processo de prototipação é mais intenso no início do processo de desenvolvimento, principalmente nas fases de concepção e pré-produção, pois o jogo ainda está tomando forma, sendo necessário definir os seus diversos componentes. Isso não impede que protótipos possam ser construídos em fases mais avançadas do projeto.

Os testes são cenários ou casos idealizados para checar se um produto funciona e/ou está de acordo com o que foi especificado nas fases de design do jogo. Isso envolve duas tarefas:

- **Validação**: checar se o produto testado realmente atende às expectativas (foi construído o que eu pedi?).
- **Verificação**: procurar por erros na execução do teste (o que eu pedi foi construído sem erros?).

Perceba que um produto pode falhar em apenas uma dessas etapas. Voltando ao exemplo do teletransporte (estou me sentindo o próprio Noturno dos X-Men!): após vários protótipos, foi decidido que o teletransporte será realizado através de locais específicos do mapa, chamados Portais de Teletransporte. Ao entrar em um portal, é aberta uma tela onde o jogador pode selecionar qualquer posição válida dentro do mapa da fase para ser transportado imediatamente.

Se na versão final do jogo, o jogador entra em um portal de teletransporte, escolhe uma posição (X,Y) e vai parar em outra (Z,W), temos um erro de verificação: a operação está de acordo com o especificado, mas não está funcionando direito! Provavelmente algum programador errou a conta na hora de calcular a nova posição do jogador (claramente, não cursou Matemática para Jogos!). Erros de verificação normalmente são bugs no sistema e o processo de correção consiste em localizar o defeito no código para reimplementar apenas a parte defeituosa.

Porém, se na versão final do jogo, o jogador chega à sala de teletransporte e na hora de escolher os locais para ir, apenas estiverem habilitados locais onde existem outros portais de teletransporte, temos um problema de validação: a operação funciona, mas não segue a especificação do projeto! Aqui o problema normalmente advém de uma falha de comunicação entre a equipe de programação e a de design, nesse caso a equipe não entendeu corretamente o que precisava ser feito. Esse tipo de erro tem o potencial de ser mais grave porque pode resultar na reimplementação de toda a funcionalidade (ou até de outros sistemas do jogo).

**Figura 04** - Erro de codificação versus Erro de especificação.


Bom, os propósitos principais dos testes são a validação das funcionalidades e verificação de erros. Mas esses não são os únicos benefícios a alcançar através da atividade de testes! Uma grande vantagem, principalmente em testes com grupos externos à equipe de desenvolvimento, é a possibilidade de **avaliar jogadores** em contato com o jogo e observar como os aspectos de jogabilidade e o conceito geral são absorvidos por eles. Se os jogadores não estiverem gostando ou se divertindo, pode ser uma boa oportunidade de descobrir quais os elementos que não estão funcionando dentro do jogo e porquê isso acontece. Ou até mesmo desistir da ideia do jogo, nos casos mais drásticos!

Outra tarefa importante que é subsidiada pelos testes é o **balanceamento** do jogo. Com o tempo, é comum que tanto a equipe de design e os testadores da casa conheçam o jogo tão bem que ele passe a se tornar fácil (depois de jogar mil vezes, até *Dark Souls* fica apenas difícil). Nesse momento pode ocorrer um fenômeno denominado ***designer blinds*** ou “viseira de desenvolvedor”: a equipe já possui um viés tão forte com relação ao jogo que começa a ter dificuldade em perceber problemas de balanceamento ou, ainda, realiza ajustes desnecessários achando que o jogo está “fácil demais”. Isso pode acarretar em um jogo muito difícil para jogadores que estarão pegando o jogo pela primeira vez, já que ele não vai ter o mesmo nível de conhecimento que a equipe a qual construiu o jogo. Ao se realizar testes com pessoas que estão interagindo com o produto pela primeira vez, os designers podem perceber esse ajuste excessivo e retornar o jogo a um estado mais balanceado de dificuldade.

Os testes estão presentes ao longo de todo o processo de desenvolvimento, porém são intensificados nas fases de produção e testes. Eles são feitos por várias equipes, desde os programadores que implementam as funcionalidades e designers que validam os sistemas produzidos até as equipes de testes, tanto internas quanto externas ao projeto.

Acho que deu para resumir a essência da prototipação e dos testes. Vamos agora detalhar um pouquinho mais sobre elas!

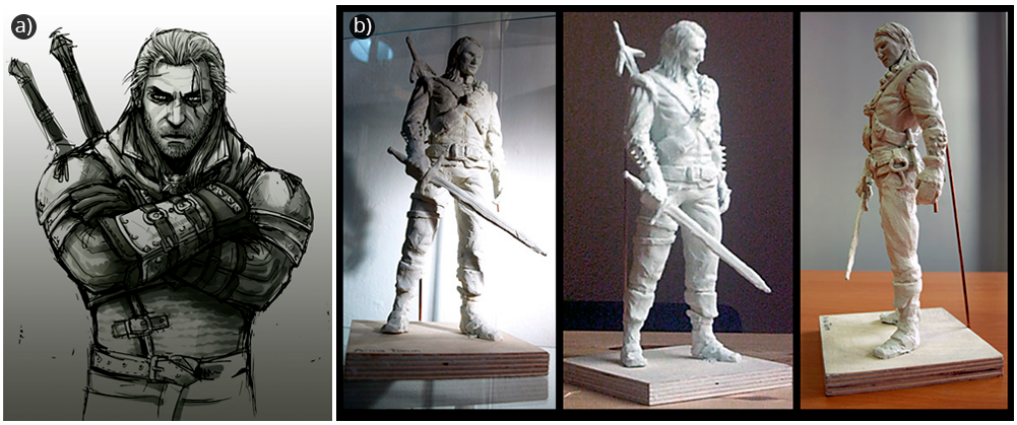
# 2 - Tipos de Protótipos

Agora você já sabe para que serve um protótipo, então veremos quais são os tipos mais comuns quando falamos de jogos digitais. Antes, vamos deixar uma coisa bem clara: um protótipo de um jogo digital não precisa ser um artefato digital!

**Figura 05** - Lá vem esse doido com as loucuras de novo.

Quer alguns exemplos? Vamos pensar aqui no desenvolvimento de um personagem para um jogo. Normalmente a equipe de arte começa com alguns desenhos de rascunho para captar a essência do personagem através de artes conceituais ou de modelos em massinha ou argila:

**Figura 06** - Exemplos de artes conceituais.
**Jogo**: *The Witcher 3*.




**Fonte**: a) http://kenji893.deviantart.com/gallery/. Acesso em 27 out. 2015.
b) https://www.deviantart.com/art/The-Witcher-Clay-Sculpture-II-99606935. Acesso em 27 out. 2015.

Realizada essa primeira definição da equipe artística, os modeladores do jogo começam a trabalhar na renderização do modelo 2D ou 3D que será utilizado, a partir de um modelo simplificado, até gerar um objeto que estará, então, pronto para ser inserido no jogo.

**Figura 07** - Modelo 3D de um personagem, sendo criado em várias etapas. Desde a construção através de vários polígonos até uma versão final, com texturas e cores aplicadas à superfície do personagem. **Jogo**: *The Witcher 3*.

**Fonte**: http://witcher.wikia.com/wiki/Character_Design. Acesso em 27 out. 2015.

Assim, o processo é finalizado com a aplicação de texturas sobre o objeto e a definição das animações para a sua movimentação. Ao final do processo temos, então, um personagem aparecendo prontinho no jogo!

**Figura 08** - Produto final do processo de design do personagem. Nada mal, hein? **Jogo**: *The Witcher 3*.

**Fonte**: http://steamcommunity.com/news/post/518257376266664639/?insideModal=1. Acesso em 27 out. 2015.

Com certeza, nessa sequência de criação tivemos protótipos digitais, como os artefatos 3D gerados em ferramentas de modelagem ou esqueletos de animação para validar a movimentação do personagem. Mas, antes de tudo isso, nós tivemos protótipos feitos em papel e plástico, que foram importantes para definir a forma e o estilo do personagem e orientaram o processo de criação até o modelo final do jogo!

Para a parte gráfica é mais fácil de visualizar o uso de protótipos analógicos, entretanto para o sistema de mecânicas? Definitivamente deve ser um protótipo digital, não é?

Não! Na verdade, é muito comum o uso de protótipos analógicos para a validação de mecânicas ou construção de níveis de um jogo. E nada melhor que um exemplo louco para mostrar como esse processo pode ser conduzido.

Imagine que você precisa testar e prototipar um jogo FPS do tipo arena, onde vários jogadores competem um contra o outro. Bem ao estilo *Counter-Strike*! E por um segundo esqueça que você consegue fazer isso clicando Novo Jogo -> Modelo FPS no seu motor de jogo. Como você poderia prototipar um jogo de FPS de forma analógica?

Primeiramente, precisamos representar o espaço físico, então você pode prototipar a jogabilidade através de um jogo de cartas, tabuleiro, etc. Para o nosso exemplo podemos desenhar um tabuleiro com o formato do nível desejado, marcando possíveis elevações com símbolos de escadas. Para cada jogador envolvido na disputa, podemos acrescentar uma peça ou token, a qual representa tanto a sua posição no tabuleiro (e eventualmente um mapa do jogo) como a direção que ele está encarando. Para finalizar, vamos discretizar o espaço do tabuleiro dividindo-o em casas de tamanho uniforme, o que facilitará a movimentação das peças ao longo das simulações.

**Figura 09** - Um exemplo muito elaborado do que pode ser feito. Isso poderia também ser simplesmente um mapa desenhado em papel A3 e balinhas Xaxá representando os personagens!

**Fonte**: https://www.hastac.org/blogs/jmsnhogan/2014/01/22/02-board-game-rapid-prototyping. Acesso em 27 out. 2015.

Ao posicionarmos todas as peças, temos no tabuleiro uma configuração espacial para o nosso jogo. Ainda podemos acrescentar peças que representem objetos dentro do jogo, como barreiras, portas e inimigos, ou podemos apenas desenhar esses objetos no mapa!

Beleza, tenho meu espaço físico. Mas como eu jogo? Bom, um jogo FPS ocorre em tempo real, mas na nossa representação de tabuleiro pode ser complicado representar essa mecânica de tempo. Teríamos que ter várias pessoas movimentando cada peça, ia ser uma loucura! Sem contar que a possibilidade de trapaça é grande! O que podemos fazer aqui?

Uma opção é discretizar o nosso tempo em unidades fixas, em que cada ação do personagem demore X unidades de tempo (que será definida por nós), como: andar uma casa do nosso tabuleiro gasta uma unidade de tempo, mudar de direção também gasta uma unidade, atirar gasta duas unidades (porque precisa mirar). Dessa forma, vamos simular os turnos de forma síncrona: cada personagem do mapa pode realizar uma quantidade fixa de unidades de tempo por turno. Assim ficamos com uma espécie de simulação câmera lenta do jogo! Se definirmos que cada jogador tem três unidades de tempo para gastar por turno, então um jogador poderia avançar três casas, mudar de direção e avançar duas casas ou atirar e avançar uma casa, por exemplo. Embora não seja uma simulação perfeita, ela nos dará uma boa ideia de como as mecânicas do jogo funcionam. Caso se deseje adicionar um elemento de incerteza a algumas ações (como o tiro), pode-se adicionar ainda uma mecânica de sorte utilizando dados para obter valores que determinam se uma ação foi bem-sucedida ou não.

Agora temos uma boa ideia de como jogar o protótipo. No entanto, ainda falta algo para tornar a simulação mais próxima da experiência de jogo real! O problema é que no jogo de FPS cada jogador visualiza apenas a sua tela, e não tem ideia do que o outro vê (menos no *GoldenEye* do Nintendo 64, que era show! E você via tudo em uma única tela dividida, então dava para fazer aquela “roubadinha”). Mas se temos apenas um tabuleiro, todo mundo conhece o estado do jogo. O que fazer?

**Figura 10** - Vou precisar de uma aspirina.

Nessa hora pegamos emprestada a experiência dos sábios jogadores de RPG: vamos usar uma entidade que controla o estado do jogo, alimenta apenas as informações necessárias para cada jogador e avalia os resultados das ações para definir o que efetivamente aconteceu no jogo. Vamos ter um *Dungeon Master* *(O Dungeon Master, ou mestre da sessão, é a pessoa que controla o andamento do jogo. Querido por todos, já que ele decide se você sobrevive ao ataque do dragão ou não!)*!

O que faremos é desenhar N tabuleiros iguais, sendo que um ficará com a pessoa que controla a sessão do jogo e N-1 ficarão com os jogadores. Cada jogador terá revelado em seu tabuleiro apenas as informações que o seu personagem sabe e consegue ver. O mestre da sessão terá um tabuleiro com todas as informações e peças dos jogadores, representando o estado completo do jogo. A partir daí o mestre controla a sessão: recolhe as ações que cada jogador toma no turno e executa-as em seu tabuleiro, determinando o resultado de cada uma. Por exemplo, se o jogador 1 está vendo o jogador 2, ele pode emitir como ações “Atirar e mover para frente”. Se o jogador 2 emitiu como ações “Mudar direção para Sul e mover duas casas”, ele será atingido pelo tiro! Se a ação do jogador 2 for “Mover 3 casas para frente”, ele escapará por um triz.

Além de realizar as movimentações, o mestre informará para cada jogador os sons que ele escuta, direção, se foi atingido ou não, se um adversário apareceu em seu campo de visão. Assim, cada jogador vai atualizando o seu próprio tabuleiro de acordo com as informações repassadas. Não é uma simulação perfeita, mas conseguimos replicar a experiência de Caça e Fuga que desejamos no jogo digital!

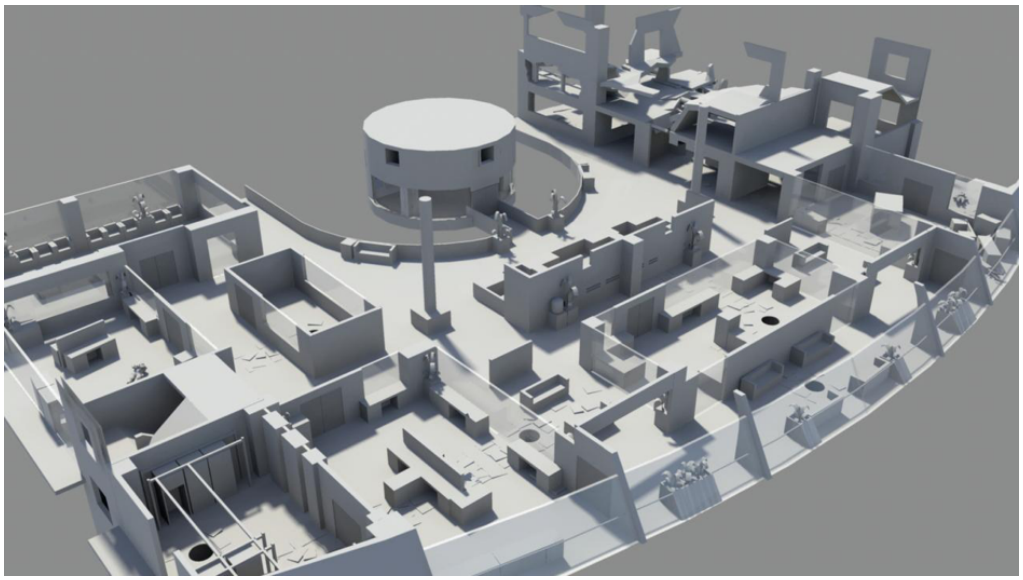
Além disso, pode-se observar alguns aspectos interessantes, como os tipos de estratégias que os jogadores empregam ou se existem pontos do mapa onde o jogador possui uma vantagem sobre os outros. E tudo isso antes de implementar a primeira linha de código ou desenhar o primeiro modelo 3D do jogo! Na verdade, essa sessão pode ser organizada em poucas horas, e em poucos dias já se pode ter uma boa ideia de como o jogo funcionará.

O uso de protótipos analógicos é extremamente interessante no desenvolvimento de jogos digitais. Porém, hoje em dia a construção de protótipos digitais é facilitada pelos motores de jogos que já possuem *templates* ou modelos de níveis e mecânicas específicos para determinados gêneros. Dessa forma, um desenvolvedor pode abrir um projeto pronto com um mapa e movimentação padrões de FPS e apenas construir os elementos necessários para a geração do protótipo desejado.

Aqui nós podemos traçar uma classificação com relação ao quanto o protótipo se aproxima da versão final do produto ou não: quando um protótipo apresenta apenas um rascunho da ideia, é chamado de protótipo de **baixa fidelidade**, e seu objetivo é entender os requisitos necessários e prover uma visualização inicial do conceito trabalhado. Normalmente os protótipos iniciais do desenvolvimento, principalmente os analógicos e os modelos digitais simplificados, se enquadram nessa categoria. Quando os protótipos estão mais próximos da representação final da ideia, dizemos que eles são protótipos de **alta fidelidade**, e já são utilizados para validação final com as equipes envolvidas ou até mesmo com o jogador (uma versão de demonstração, por exemplo). Esses protótipos também são utilizados para apresentar partes do jogo para a comunidade como estratégia de marketing.

Dentre os protótipos de baixa fidelidade, destacamos um que é usado de forma recorrente no desenvolvimento de jogos: o nível de caixa cinza ou ***Gray-box Model***. Esse protótipo é construído com o intuito de testar a jogabilidade, sistemas de mecânicas ou navegação em um determinado nível do jogo, e tem como principal caraterística a ausência de artefatos gráficos em versão final. Ele normalmente é construído utilizando-se elementos geométricos padrões, como retângulos e caixas, apenas para simular obstáculos e elementos de interação com o jogador, dando a aparência de um jogo construído com caixas de papelão ou caixas cinzas (por isso ele tem esse nome).

**Figura 11** - No protótipo ***Gray-box Model***, o importante é o teste da jogabilidade e das mecânicas do jogo.



**Fonte**: http://pete-ellis.com/kzm/. Acesso em 27 out. 2015.

Do lado dos protótipos de alta fidelidade, destacamos o **Vertical Slice** ou lâmina vertical. Esse tipo de protótipo é feito com o objetivo de apresentar o jogo e é implementado como uma pequena porção completa (algumas vezes menor do que um nível), contendo artefatos e sistemas em suas versões finais ou mais atuais de desenvolvimento. A ideia é passar uma visão de como será o jogo final, podendo o protótipo ser interativo (com um espaço e/ou conjunto limitado de ações) ou apenas expositivo, como uma filmagem em vídeo das principais características do jogo.

**Figura 12** - Uma *vertical slice* objetiva mostrar como todas as camadas de um jogo se combinam, dando uma ideia da versão final. Porém ela é feita em um escopo limitado, não explorando todos os aspectos do jogo.

**Fonte**: http://askagamedev.tumblr.com/post/77406994278/game-development-glossary-the-vertical-slice. Acesso em 27 out. 2015.

Esses não são os únicos tipos de protótipos, mas já são uma boa introdução! E qual eu devo usar? Tudo depende do seu projeto, afinal cada jogo é único e demanda técnicas diferentes. Mas existem algumas dicas que podem ajudar você no uso e na escolha dos protótipos. Vamos a elas!

# 3 - Boas práticas para prototipação

---

Nessa seção vamos listar algumas dicas que autores e designers da indústria consideram chaves para se ter um processo de prototipação eficiente.

- Saiba exatamente **porque** você está construindo o protótipo. Parece besteira, mas não é. O objetivo do protótipo ajuda a definir qual será o tipo, o escopo, a mídia em que será construído e até mesmo a qualidade (se será de baixa ou alta fidelidade). A ideia é apresentar algum aspecto do jogo para os jogadores? Você deseja validar uma mecânica com a equipe de design em uma reunião de *brainstorm*? Deseja saber se é possível implementar um sistema de personagens inteligentes com comportamento de grupos no motor utilizado pela equipe de programação? Deseja saber se um determinado estilo visual agrada ao seu público-alvo? Saber qual pergunta se deseja responder é essencial para construir um protótipo que efetivamente contribua para o processo de desenvolvimento.

- **Não se apegue demais ao seu protótipo**! A ideia da prototipação rápida é construir vários produtos que possam ser utilizados, validados e **descartados** de acordo com a necessidade. Apesar de você ter investido um tempo construindo o protótipo, isso não significa que ele deve ser obrigatoriamente utilizado no projeto. Muitos programadores ou designers sentem que descartar um protótipo é o mesmo que descartar horas de trabalho, mas isso não é verdade. Muitas vezes o protótipo serve apenas para que você refine sua ideia, e a visão que o protótipo lhe dá é algo que não se perde, mesmo se você der um delete nele!

- Procure organizar a sua construção de protótipos de acordo com a **prioridade** para o projeto. No caso de um RPG voltado para a ação, é importante iniciar com a prototipação dos sistemas de combate e movimentação e depois partir para sistemas secundários, como produção de itens ou equipamento. Na medida do possível, tente paralelizar a produção dos protótipos entre as diversas equipes disponíveis: enquanto a equipe de programação implementa o sistema de combate, a equipe de artes vai criando os desenhos conceituais e cenários, e a equipe de narrativa rascunha uma sinopse da história e dos personagens do jogo.

- Um bom caminho para iniciar a construção de um jogo incide em criar um ambiente de interação livre onde as mecânicas possam ser utilizadas. Muitos designers chamam isso de "*build the toy first*" ou construir o brinquedo primeiro. A ideia dessa abordagem refere-se a construir um ambiente e perceber como as mecânicas funcionam nele, para se ter melhor ideia do que seria divertido fazer no jogo e, assim, traçar objetivos mais expressivos para o jogador. Muitos jogos utilizaram essa abordagem, como o GTA: primeiro a equipe construiu uma cidade e várias mecânicas de interação para, depois, definir qual seria a temática do jogo e os objetivos que o jogador teria de alcançar.

Uma dica final seria com relação ao uso de ferramentas as quais permitem mudanças em tempo real dentro do projeto. Se toda vez que um membro da equipe de design quiser fazer uma mudança (por exemplo, colocar uma armadilha mais adiante da fase para ver se o nível fica mais fácil/difícil), for necessário executar o projeto inteiro, o processo de validação dos protótipos vai demorar muito. É comum encontrar ferramentas de edição em que o designer pode realizar alterações em tempo de execução (ele muda o valor de uma variável e a mudança ocorre na tela), isso agiliza o processo e permite uma maior flexibilidade e independência das equipes de desenvolvimento. Caso o motor do jogo não tenha esse tipo de suporte, é interessante solicitar que a equipe de programação construa uma ferramenta como essa, que possa ser disponibilizada tanto para a equipe de artes como para a equipe de design.

# 4 - Testando 1, 2, 3...

A atividade de testes ocorre durante todo o processo de desenvolvimento de um jogo. O objetivo geral é garantir a qualidade do jogo, tanto no aspecto técnico como no quesito diversão.

Uma das atividades iniciais de entrevista e pesquisa com os jogadores, que pode ocorrer no início do desenvolvimento de um jogo, é a formação de **grupos focais**. O objetivo é tentar validar no mercado se uma ideia de jogo é realmente interessante, ou inverter o processo de design na tentativa de idealizar um jogo a partir do perfil e das necessidades expressas pelos entrevistados. Essa abordagem deve ser realizada de maneira cautelosa, pois se for feita com um grupo pequeno, é provável que não exista uma representação real do mercado em geral.

A dificuldade de se realizar esse tipo de abordagem está na escala em que se deseja realizar a pesquisa: caso o objetivo seja um grande número de respostas, pode-se utilizar um formulário online, mas a qualidade das respostas é inferior a de um método como entrevistas com pessoas ou grupos. Porém o custo para se realizar entrevistas com um grande número de pessoas pode se tornar proibitivo, principalmente para projetos menores. Essa metodologia é aplicada normalmente quando não se tem ainda uma ideia certa do jogo que se deseja desenvolver.

Ao longo do processo de desenvolvimento existem vários tipos de testes que podem ser realizados para avaliar a qualidade do produto:

- **Testes de usabilidade** verificam se a interface com o jogador está intuitiva e respondendo bem aos comandos, se o esquema de navegação não é complexo demais e se os controles são confortáveis. Além disso, a interface pode ser testada em função dos seus aspectos visuais, como disposição dos elementos e esquema de cores utilizado.

- **Testes de plataforma** verificam se o jogo se comporta de maneira esperada em cada uma das plataformas-alvo para as quais o projeto é desenvolvido, verificando questões pertinentes ao hardware e à compatibilidade de funcionalidades e interfaces de controle.

- **Testes de localização** verificam se as traduções para outros idiomas estão adequadas e mantendo a coerência dos textos originais.

Para cada sistema de mecânicas que criamos, é necessário realizar uma bateria de testes para validar se eles funcionam de forma individual e de forma integrada com os outros sistemas do jogo.

Apesar de variados, existem dois tipos de testes os quais são predominantes: os **testes de software** e os **testes de jogabilidade**, que ocorrem ao longo de todo o processo e englobam alguns dos testes mencionados acima. Vamos detalhá-los nas subseções a seguir!

## 4.1 - Teste de Software

Os testes de software são desenvolvidos pela equipe de programação e têm como objetivo verificar e validar o código desenvolvido para o jogo. Ou seja, se ele está correto e se efetivamente implementa o que foi especificado para cada funcionalidade.

Testar um software é um processo que ocorre em diversas camadas: desde os componentes básicos que formam o programa (classes, funções) até os sistemas inteiros do jogo (jogabilidade principal, sistemas de IA, sistemas de renderização gráfica, sistemas de combate, etc.) e, por fim, o jogo como um todo. Esses podem ser realizados pelo próprio desenvolvedor das funcionalidades, mas é comum existir uma equipe de testes específica para avaliar o código produzido. Dessa forma, se tem olhares diferentes sobre uma mesma parte do jogo, o que facilita a detecção de erros ou equívocos do programador.

Com relação à abordagem do teste, podemos classificá-los em dois tipos: testes de caixa preta e testes de caixa branca.

**Testes de caixa preta** ou **comportamentais** são testes baseados na especificação da funcionalidade, e não na sua estrutura. Dessa forma, o testador executa vários cenários possíveis sem observar a forma como a funcionalidade foi codificada, e o caso de teste ocorre com a alimentação de diversos valores de entrada, os quais geram uma saída que é comparada com o resultado esperado a partir da especificação da função.

Imagine um jogo de tabuleiro 2D, com uma perspectiva de câmera *top-down*, onde o personagem tem uma posição representada por dois valores (X, Y). Considere também que o canto inferior esquerdo representa a coordenada (0, 0) no tabuleiro e que seu tamanho máximo é de (512, 512). Logo, uma especificação para a função *mover_no_tabuleiro* do jogo poderia ser dada da seguinte forma:

- Direcional para cima faz o valor do Y aumentar uma unidade por vez.
- Direcional para baixo faz o valor de Y diminuir uma unidade por vez.
- Direcional para a direita faz o valor de X aumentar uma unidade por vez.
- Direcional para a esquerda faz o valor de X diminuir uma unidade por vez.
- O personagem não pode sair dos limites do tabuleiro.

Com essa especificação, nós podemos criar vários casos de teste sem precisar verificar como o código da função *mover_no_tabuleiro* foi construído! Basta fornecer uma sequência de movimentações, calcular a saída esperada e verificar se a função retornou o mesmo valor. Se o jogador está na posição (2, 3) e ele anda duas vezes para a direita e uma para cima, sua posição deve ser (4, 4). O problema é que testar todas as possibilidades é impossível, dado que o conjunto de combinações diferentes que podem ser executadas é praticamente infinito! Para isso, nós vamos precisar definir um conjunto de caso de testes que seja representativo o suficiente para cobrir as diversas possibilidades de cenários válidos e inválidos que poderiam ocorrer dessas simples regras de movimentação.

Uma forma de realizar isso é através da criação de classes de equivalência, ou seja, cenários válidos e inválidos que nosso programa pode encontrar ao longo de sua execução. Considerando o exemplo da movimentação, uma classe válida seria uma movimentação para cima que aumenta apenas o valor do componente Y e não sai do tabuleiro. Se testarmos alguns casos com esse tipo de movimentação e eles operarem de forma correta, é provável então que ela funcione para todos as outras movimentações dessa mesma classe (movimentações para cima, dentro do tabuleiro).

Já se considerarmos um caso limite (o personagem na borda de cima do tabuleiro), então teríamos uma classe inválida se o personagem se movimentasse além das bordas. Seguindo esse raciocínio, podemos construir um conjunto relativamente pequeno de casos de testes que cubram todos os tipos de movimentação existentes no tabuleiro sem ter de, necessariamente, testar todas as possibilidades de forma exaustiva!

**Figura 13** - Exemplos de casos de teste para o tabuleiro usado como exemplo.


**Testes de caixa branca** ou **estruturais** têm como objetivo avaliar a estrutura do código e verificar se todos os caminhos possíveis de execução do programa são testados e verificados. Através desse teste é possível encontrar partes de código "mortas" (nunca são atingidas pelo programa), o que pode indicar erros de implementação ou a necessidade de refatoração (reescrita, falado de forma chique) do código da funcionalidade para eliminar partes obsoletas. Um problema do teste de caixa branca é que, como o testador vê a estrutura do código, ele acaba sendo condicionado a escrever testes os quais se adequem àquela estrutura, não pensando em casos que possam verificar se aquele código cobre todas as possibilidades da especificação.

Normalmente, os dois tipos de teste são aplicados em conjunto, já que possuem finalidades diferentes, mas com caráter complementar para validação de um programa ou componente do sistema.

Outra classificação que podemos realizar com relação aos testes é em função do escopo no qual o teste é executado. Testes que são executados nos elementos mais básicos do sistema (como classes, componentes e funções) são chamados de testes

unitários. Esses testes buscam validar cada unidade em relação às entradas e saídas definidas na interface do componente, e tem como objetivo garantir que cada unidade funciona corretamente.

O teste unitário é feito apenas com o componente específico, para evitar que erros provenientes de outras partes do sistema possam se propagar e resultar em efeitos colaterais no componente testado (tipo efeito borboleta e teoria do caos, só que o tufão acontece no seu trabalho). Esse tipo de teste é utilizado no processo de desenvolvimento para garantir que, mesmo após mudanças no código do sistema, seus componentes ainda funcionem corretamente. Uma outra vantagem é que erros detectados estão relacionados diretamente ao componente testado, facilitando a sua localização e correção. Além disso, se o programador já pensa nos casos de teste que serão implementados, é mais provável que ele avalie bem a especificação da funcionalidade e entenda melhor o que deve fazer, reduzindo erros decorrentes de interpretação equivocada.

Por fim, mas não menos importante, existem vários frameworks que automatizam os testes unitários! Dessa forma, o programador implementa o caso de teste como um programa e a tarefa maçante e repetitiva de executá-lo várias vezes fica a cargo do computador! Isso é interessante também do ponto de vista de manutenção de código: quando ocorrer uma mudança no sistema, basta colocar todos os casos de teste para rodarem novamente e verificar se as mudanças efetuadas geram algum erro no programa.

Além dos testes unitários, existem mais dois tipos de testes que devem ser realizados para garantir o funcionamento de um software. Os **testes de integração** verificam se não há falhas operacionais dos componentes quando utilizados de forma conjunta. Embora cada componente possa ter passado por testes de forma individualizada, ainda podem ocorrer erros no momento da comunicação entre dois componentes diferentes, logo esse teste visa detectar erros na especificação da interface entre partes do programa. Em um último nível, o jogo será testado de forma completa, com um **teste de sistema**: da mesma forma que os componentes individuais são agrupados em módulos ou subsistemas para o teste de integração, esses módulos são agrupados e executados em conjunto para verificar o funcionamento do sistema como um todo. Esse tipo de teste serve para verificar funcionalidades e características de desempenho e usabilidade do jogo de modo geral.

Um último tipo de teste comum em sistemas de software é o **teste de aceitação**, e refere-se a um aceno do usuário final, indicando que o sistema corresponde às suas expectativas. No entanto, jogos não são sistemas de softwares comuns, e os testes de aceitação costumam ser executados com vários grupos distintos, em momentos diferentes do ciclo de desenvolvimento. Esses são os chamados testes de jogabilidade, os quais veremos a seguir!

## 4.2 - Testes de Jogabilidade

Se o teste de software fica a encargo dos programadores, o teste de jogabilidade é uma ferramenta essencial para os designers! Através dele serão observados vários elementos do jogo: seu balanceamento, a forma como os jogadores abordam o jogo, as estratégias desenvolvidas, o engajamento durante as sessões de jogo e, o mais importante, se o jogo é divertido ou não!

**Figura 14** - Opa, acho que o jogo não estava tão bom!

O primeiro passo para a realização do teste é definir o grupo de testadores. Ao longo do projeto esse grupo vai alternando: inicialmente, os designers e a equipe de desenvolvimento realizam os primeiros testes com o intuito de verificar como os sistemas do jogo estão progredindo. A partir dessas primeiras impressões, já podem ser necessários ajustes, fazendo com que o jogo fique cada vez mais refinado. O problema é que se uma mesma equipe testa repetidamente o mesmo jogo (lembra o que falamos sobre a viseira do desenvolvedor?), esse olhar “mal-acostumado” dos

desenvolvedores pode evitar que eles percebam um problema com o jogo, ou pior, levar a ajustes desnecessários devido a uma percepção errônea da dificuldade ou balanceamento do jogo.

**Figura 15** - Esse jogo está muito fácil!
**Jogo**: *Dark Souls*


**Fonte**: http://www.giantbomb.com/dark-souls/3030-32697/forums/durante-from-neogaf-fixes-resolution-problems-558179/. Acesso em 28 out. 2015.

Nesse momento, é importante que haja o envolvimento de outras equipes para testar o jogo. Um primeiro passo é contratar testadores experientes, de preferência jogadores hardcore do gênero específico do jogo que está sendo desenvolvido. Esses jogadores possuem uma carga de conhecimento de outros jogos que pode estabelecer uma base de comparação com o seu, e o feedback deles é essencial nessa fase intensa de ajustes. À medida que o jogo se aproxima de uma versão alfa, é importante demonstrá-lo a novos testadores para que eles tenham um primeiro olhar sobre o jogo. Isso permite sempre uma perspectiva nova sobre o jogo e seus elementos, aumentando as chances de criar uma experiência balanceada para os jogadores.

Um segundo aspecto que deve ser observado é com relação ao local do teste. Ele pode ser realizado em um local físico específico ou na própria casa dos testadores, via internet. O teste *in-loco* (na própria empresa ou em local reservado para essa atividade) permite que a equipe de design observe os jogadores em ação e colha dados diretamente da sessão de testes, porém tem uma escala limitada, devido à restrição física de quantos testadores podem ser alocados para realizar o

teste. Já com a execução online é possível obter uma grande massa de testadores, porém a coleta de dados é dificultada. Normalmente se aplicam questionários (quase sempre respondidos de forma “automática” pelo jogador) ou pela coleta de dados da sessão de jogos de cada testador, sendo feita uma mineração em cima dessa base para tentar descobrir padrões de comportamento. É importante que os jogadores os quais participam do teste tenham total ciência sobre esses fatos, pois o uso de informações pessoais sem consentimento pode acarretar em sérios prejuízos legais para o desenvolvedor do jogo!

Assim como os protótipos, os testes são realizados com um objetivo específico em mente. Um teste pode ser realizado para avaliar apenas o sistema de controles do jogador ou a navegação de um determinado nível. Em fases finais do processo de desenvolvimento, os testes enfatizam na avaliação da jogabilidade geral e do balanceamento do jogo.

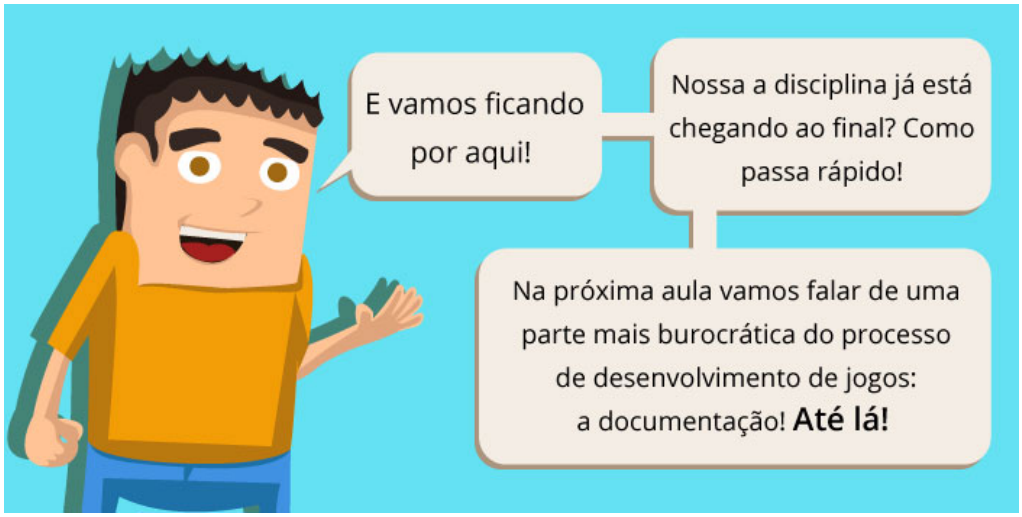
Por fim, é muito importante que o designer se atenha a uma determinada metodologia no momento de realizar os testes, pois ele precisa decidir como procederá em três etapas do teste:

- **Preparação inicial**: o designer deve decidir o quanto de informação precisará fornecer ao jogador antes de iniciar a sessão de teste. Dependendo do foco do teste, ele pode fornecer uma breve informação sobre os esquemas de controle e/ou sobre o nível que o jogador testará. Em testes com versões finais do jogo o ideal é não fornecer nenhuma informação e verificar se o jogo consegue ensinar e informar o jogador de forma adequada com relação às mecânicas e à narrativa, e se o jogador consegue perceber os objetivos de forma clara.

- **Observação do jogador**: aqui é extremamente importante que o designer tenha uma postura passiva! Você nunca deve interferir com uma sessão de jogo ou tentar orientar o jogador sobre uma forma “correta” de interação. Se o jogador não percorre um nível ou enfrenta um desafio do jeito que você planejou, tente observar porque isso aconteceu, em vez de dizer que ele deveria ter ido para o outro lado. Isso possibilita que você altere o seu jogo para melhor orientar o jogador, ou ainda pode revelar possibilidades de jogabilidade (principalmente emergente), as quais você ainda não havia percebido!

- **Coleta de dados**: ao longo da sessão de jogos, é importante para o designer coletar dados sobre o que o jogador está fazendo (que decisões ele toma, que caminhos ele segue no jogo, por exemplo), podendo ser através de filmagens da sessão de jogo, com câmeras focando as reações do jogador e a tela do jogo. Isso pode revelar em quais momentos o jogador está sentindo maior tensão, em quais momentos ele está entediado, quais os momentos que o divertiram mais, etc. Esse tipo de informação pode ser essencial nos ajustes finais! Caso não seja possível a captura da sessão ao vivo, o uso de entrevistas e questionários para entender a experiência que o jogador vivenciou também consistem em ferramentas importantes para adquirir conhecimento sobre o que funciona ou não dentro do jogo.

Seguindo esses passos, o designer aumenta a chance de conseguir informações produtivas e de criar um jogo realmente divertido e engajante! Claro que representam mais dicas do que procedimentos obrigatórios: o processo pode se adequar à realidade do seu projeto ou empresa. O importante é focar nos objetivos principais do teste de jogabilidade e entender como o jogador percebe o seu jogo, quais elementos o divertem e quais elementos não estão funcionando.




## Pontos-chave

Nossa aula 2 em 1! Os assuntos dessa semana fecham a nossa exploração da tétrade elemental, mas isso não quer dizer que vocês viram tudo o que há sobre o assunto. Aproveitem para dar uma olhada na sessão de leitura complementar ou, se

possível, consultar as fontes presentes na seção de referências. Mas antes disso, que tal reforçarmos um pouco o que vimos hoje?

- Protótipos são produtos que representam uma versão simplificada do projeto (ou parte dele).

- Protótipos possuem um escopo bem definido e são construídos com um objetivo específico em mente. São alinhados à ideia de ciclos rápidos de iteração do processo de design.

- Os protótipos podem ser desenvolvidos em mídias analógicas ou digitais. Com relação à qualidade, eles podem ser de Baixa fidelidade (*Gray-box Model*) ou Alta fidelidade (*Vertical Slice*).

- Testes são cenários e casos utilizados para efetivar a validação e verificação do jogo desenvolvido. Também permitem que os designers observem a interação dos jogadores com o jogo, auxiliando nos ajustes e no balanceamento.

- Ao longo do desenvolvimento de um jogo, existem vários tipos de testes que são aplicados para averiguar a qualidade da produção, como testes de usabilidade, plataforma, software e jogabilidade.

- Os testes de software podem ser classificados com relação ao objetivo e à metodologia, como testes de Caixa Preta (Comportamentais) ou teste de Caixa Branca (Estruturais).

- Os testes de software podem ser classificados com relação ao escopo do teste em: unitários (componentes básicos do sistema), integração (conjunto de componentes, módulos) e sistema (jogo completo).

- O teste de jogabilidade têm como objetivo avaliar a interação do jogador com os elementos do jogo, se o balanceamento está adequado e se o jogo está divertido.

# Leitura Complementar

Olha, esta semana caprichamos! Às vezes eu penso que essa seção de leitura complementar é a melhor parte da aula. Tantos materiais legais para ler/assistir! Confiram abaixo:

- Um relato de experiência de prototipação de um jogo, com várias dicas e comentários sobre as decisões tomadas no processo de construção do jogo.
- Um artigo breve comentando quais as principais diferenças entre o processo de teste de softwares tradicionais e os testes de jogos digitais.
- Uma matéria interessante mostrando que nem tudo são flores na vida dos profissionais da área de testes de jogos. Uma leitura para conscientizar sobre as dificuldades enfrentadas na área!
- Um conjunto de minijogos e protótipos que foram concebidos durante o desenvolvimento do jogo *Spore*. Esse repositório mostra o quanto o investimento em prototipação é necessário para a criação de bons jogos, principalmente quando a ideia trabalhada é inovadora!

# Autoavaliação

1. Quais problemas podem existir quando não usamos prototipação e testes no nosso projeto?
2. A utilização de testes automatizados permite um aumento de produtividade no projeto a longo prazo. Quais tipos de testes podem usufruir dessa técnica?
3. Qual tipo de protótipo seria melhor utilizar para vender a ideia do jogo a uma empresa? E qual tipo de protótipo seria melhor para discutir ideias sobre um sistema de stealth para um jogo?

# Referências


ASURA THE GAME. **Grey boxing levels to save your life and time**! Disponível em: <http://asurathegame.com/blog/grey-boxing-levels-to-save-your-life-and-time/>. Acesso em: 25 out 2015.

GAMES FROM WITHIN. **Prototyping**: you're (probably) doing it wrong. Disponível em: <http://gamesfromwithin.com/prototyping-youre-probably-doing-it-wrong>. Acesso em: 24 out. 2015.

GRAY, Kyle. et al. **How to prototype a game in under 7 days**. Disponível em: <http://www.gamasutra.com/view/feature/130848/how_to_prototype_a_game_in_under_7_.php?print=1>. Acesso em: 25 out. 2015.

HECKER, Chris. **Advanced prototyping**. Disponível em: <http://chrishecker.com/Advanced_Prototyping>. Acesso em: 26 out. 2015.

NAYAK, Soubhagya. **Advantages of Unit Testing**. Disponível em <https://moanubhuti.wordpress.com/2011/04/23/advantages-of-unit-testing/>. Acesso em: 27 out. 2015.

ROGERS, Scott. **Level up!** The guide to great video game design. John Wiley & Sons, 2014.

SCHELL, Jesse. **The Art of Game Design**: A book of lenses. CRC Press, 2008.

SOMMERVILLE, Ian. **Software engineering**. 9. ed. Addison Wesley, 2011.

WHAT GAMES ARE. **Vertical slice**. Disponível em <http://www.whatgamesare.com/vertical-slice.html>. Acesso em: 26 out. 2015.