

# GamEXP: Um Modelo para Balanceamento de Elementos da Mecânica em Jogos Digitais

Ariel F. Bello<sup>1</sup>, Vinícius J. Cassol<sup>2</sup>

<sup>1</sup>Faculdade de Sistemas de Informação EaD  
Universidade do Vale do Rio dos Sinos (UNISINOS) – São Leopoldo – RS

<sup>2</sup>Faculdade de Jogos Digitais  
Universidade do Vale do Rio dos Sinos (UNISINOS) – Porto Alegre – RS

{arielfbello, cassol.vinicius}@gmail.com

**Abstract.** *Player participation in game development is increasingly prominent. However, many contributions can occur only through the game's team mediation. In this context, a model is presented in which players can create and test new mechanics configurations, whilst generating data for balancing this element. The model is then tested through the implementation of a game, a server, data gathering and analysis according to its rules. Finally, game designers are surveyed, showing their positive understanding about the work's relevance.*

**Resumo.** *A participação dos jogadores no desenvolvimento de jogos está cada vez mais eminente. Porém, muitas dessas interações só acontecem com a intermediação da equipe do jogo. Neste contexto, se apresenta um modelo onde os jogadores podem criar e testar novas parametrizações de mecânica, ao mesmo tempo em que geram dados para o seu balanceamento. O modelo é então posto em teste através de um jogo, um servidor, coleta e análise, conforme previsto. Após avaliar os resultados, conclui-se que o modelo é funcional. Por fim, um questionário acerca do modelo é realizado com game designers, onde eles mostram uma percepção positiva quanto a relevância do mesmo.*

## 1. Introdução

A criação de um jogo é uma tarefa desafiadora, envolve uma ideia que deve ser trazida à vida com sons, imagens, tecnologia, roteiro e interatividade. Todos estes elementos juntos formam uma unidade que provê uma experiência específica ao jogador. Para que esta experiência seja agradável, estes elementos devem estar balanceados.

Em um cenário em que a participação dos jogadores no desenvolvimento vem crescendo [Sanders and Stappers, 2008], ainda é muito comum eles terem um papel bem restrito e definido em uma iteração de balanceamento: testam e opinam. Porém a iteração completa engloba o desenvolvimento do protótipo com respectivos testes, análise e refinamento [Fullerton, 2008]. Além disso, a nova versão (refinamento e novo protótipo decorrente) que é resultado da análise dos dados obtidos através do jogador, não é desenvolvida por ele, impedindo assim que este (e outros) possam experimentar diretamente seu conceito.

Os jogadores têm um grande potencial criativo, sua opinião a respeito da experiência que um jogo oferece é relevante, e desempenham um papel essencial no processo de

balanceamento. Este trabalho propõe um modelo, aqui chamado GamEXP, que tem como objetivo explorar este contexto. Neste modelo, jogadores podem parametrizar variáveis da mecânica, testar e avaliar diretamente as configurações criadas, sem intervenção do *game designer* e/ou sua equipe. Estas interações geram dados que são agregados e disponibilizados por um servidor para que o *game designer* tenha informações relevantes na escolha dos valores definitivos para estas variáveis.

O artigo está organizado em sete seções. A primeira Seção apresenta o contexto do trabalho e seu objetivo. Na segunda Seção são apresentados conceitos importantes para este trabalho. A terceira Seção discute a participação dos jogadores no desenvolvimento de jogos. O modelo é apresentado na quarta Seção e na quinta é descrito um caso de sua implementação. A sexta Seção faz uma avaliação do modelo com os dados coletados ao longo do projeto. Por fim, a sétima Seção apresenta as considerações finais.

## 2. Fundamentos

Esta seção apresenta conceitos importantes na discussão deste trabalho. Serão explorados mecânica de jogo; jogabilidade; balanceamento, com suas diferentes formas; e testes, com seus respectivos tipos.

### 2.1. Mecânica de Jogo

Jogos são uma forma de entretenimento, assim como quadrinhos, filmes e livros, mas existe um elemento que pertence exclusivamente aos jogos, e é justamente este que os distingue das demais: a sua mecânica [Schell, 2008]. Isto decorre da classificação que o autor faz dos constituintes dos jogos: estética, roteiro, tecnologia e mecânica. Enquanto os três primeiros são compartilhados com mídias lineares de entretenimento, o último é essencial à interatividade e não pertence a elas.

Mecânica são as regras, processos e dados no coração de um jogo [Adams and Dormans, 2012]. Ela define o progresso, o que acontece em que momento e os objetivos do jogo [Adams and Dormans, 2012] [Schell, 2008].



Figura 1. Mecânica de pulo no jogo Super Mario World. Fonte: Nintendo [1990]

Um exemplo de mecânica é o pulo do personagem Mario no jogo Super Mario World [Nintendo, 1990] (Figura 1). Neste caso, a mecânica está associada a mais de uma regra: a força do pulo e a condição de poder iniciar somente quando o avatar estiver com os pés no chão.

## 2.2. Jogabilidade (*Gameplay*)

Jogabilidade é a interação formalizada que ocorre quando jogadores seguem as regras de um jogo e experienciam seu sistema ao jogá-lo [Salen and Zimmerman, 2004]; Adams e Dormans [2012] a definem como os desafios que o jogo apresenta ao jogador e as ações que o jogador pode realizar.

Nota-se que jogabilidade é um elemento que surge da interação do jogador com as regras definidas pela mecânica. Este é um conceito mais subjetivo, contudo é um termo muito utilizado e importante ao se avaliar um jogo.

## 2.3. Balanceamento

Balanceamento, segundo Schell [2008], é o ajuste dos elementos do jogo até se alcançar a experiência desejada. “Um *chef* virtuoso pode fazer da mais simples das receitas uma delícia, assim como um bom *game designer* pode tornar o mais simples dos jogos um prazer de jogar — ambos sabem como equilibrar os ingredientes” [Schell, 2008, p. 171].

Este processo certamente terá objetivos muito diferentes a depender do jogo, por exemplo, em um jogo *multiplayer*, equilíbrio quer dizer que as posições iniciais e as jogadas são justas para todos. Em jogos *single-player*, quer dizer que o nível de habilidade está bem ajustado para o público alvo [Fullerton, 2008].

Existem diferentes classificações para as diferentes categorias de balanceamento. Será discutida a categorização de Fullerton [2008] em quatro áreas: variáveis, dinâmica, condições iniciais e habilidade.

As variáveis são números que definem propriedades de elementos do jogo. Tomando novamente Super Mario World como exemplo, uma variável poderia ser a velocidade de um Koopa Troopa (tartarugas inimigas de casco vermelho). Se sua velocidade fosse dez vezes a velocidade do Mario, os Koopas seriam um desafio muito difícil, por outro lado, se fosse tão baixa a ponto de ficarem quase parados, não apresentariam um desafio substancial. Neste caso, a velocidade deste inimigo um pouco abaixo da velocidade do avatar do jogador apresenta uma experiência balanceada.

A dinâmica se refere às situações emergentes quando o jogo está em pleno funcionamento. Em um *Role Playing Game* (RPG) como Fallout [Interplay, 1997], em que se pode escolher os atributos do personagem e existem diversas maneiras de progredir, por vezes jogadores descobrem combinações de atributos e/ou habilidades específicas que permitem completar o jogo com muita facilidade<sup>1</sup>. Estas formas de interação são chamadas de estratégias dominantes [Fullerton, 2008]. Situações como essa não são criadas intencionalmente e procura-se evitá-las.

---

<sup>1</sup>Normalmente leva-se várias horas para terminar este jogo. Todavia o vídeo disponível no link [https://www.youtube.com/watch?v=WzSOKi\\_t5fg](https://www.youtube.com/watch?v=WzSOKi_t5fg) mostra um *Speed run* onde o jogo é completado em 9 minutos e 19 segundos.

Equilibrar as condições iniciais significa garantir que o sistema dá as mesmas chances de vitória a todos os jogadores. Alguns jogos são simétricos justamente para solucionar este problema, como o Xadrez, em que ambos os jogadores têm as mesmas 16 peças e o elemento assimétrico de um começar jogando não gera vantagem significativa. Em jogos assimétricos, como em um enfrentamento de dois jogadores de raças diferentes no Starcraft II [Blizzard, 2010], as raças são balanceadas de tal forma que, apesar das diferenças, as chances de ganhar dependem da habilidade de cada jogador.

Por último, balanceamento de habilidade é nivelar a dificuldade do desafio à habilidade do jogador. Muitos jogos resolvem este problema disponibilizando níveis variados de dificuldade.

## 2.4. Teste de jogos

De maneira geral, testes são fundamentais para construção de software e isto não é diferente para jogos. Schell [2008], Fullerton [2008], Salen e Zimmerman [2004] descrevem o processo de desenvolvimento de jogos como iterativo, no qual os testes desempenham um papel fundamental. Existem testes diferentes, com diferentes propósitos. Esta seção descreve quatro tipos de testes: grupos focais, *Quality Assurance (QA)*, usabilidade e *playtest*, com destaque para o último, que será o tipo de teste aplicado neste trabalho.

Grupos focais são formados por alguns representantes (normalmente de 6 a 12 usuários em potencial) de um grupo específico, que se juntam à parte interessada para discutir e opinar sobre questões pertinentes a equipe de design do jogo. São discutidas quais funcionalidades são importantes para os consumidores e o que eles pensam sobre vários conceitos do projeto. Geralmente são utilizados *storyboards* para mostrar os conceitos do jogo para o grupo [Davis et al., 2005].



**Figura 2.** Uma criança com aspecto peculiar em *The Sims 3*. Fonte: Eletronic Arts [2009]

Testes de *QA*, no português garantia de qualidade, procuram encontrar erros e garantir a funcionalidade esperada. Para Schell [2008, p. 390], “Este teste não tem nada a ver com quão divertido o jogo é, e tudo a ver com procurar *bugs*”. Portanto, estes testes

devem encontrar, antes dos jogadores, *bugs* como dragões voando para trás aleatoriamente em Skyrim<sup>2</sup>, ou uma criança com partes do corpo sendo esticadas de uma hora para outra em The Sims 3 [Eletronic Arts, 2009] (Figura 2), dentre outros exemplos<sup>3</sup>.

Testes de usabilidade têm o objetivo de validar se o sistema e a interface de jogo são intuitivos e fáceis de usar. Ambos são necessários para uma boa experiência, porém não são suficientes [Schell, 2008]. Os testes de usabilidade buscam mostrar se um jogo é acessível, monitorando os movimentos e as respostas dos jogadores, seja por câmeras, questionários, observação direta e/ou outros tipos de sensores. É importante garantir que os novos jogadores consigam navegar facilmente pela interface, aprendam os controles e os empreguem de maneira efetiva, assim como devem entender os desafios propostos e as ações disponibilizadas para que eles possam superá-los.

O *playtest* é realizado durante todo o processo de *design* para tentar avaliar se o jogo está ou não atingindo os objetivos almejados para a experiência do jogador [Fullerton, 2008]. Este é o tipo de teste mais intrínseco aos jogos, pois é o que observa a interação do usuário com este em funcionamento. Aqui surge a experiência criada pelo jogo e é o momento de tentar avaliar aspectos complexos (e muito importantes) como jogabilidade. Para obter melhores resultados de um *playtest*, segundo Schell [2008], é necessário responder às questões “Por quê?”, “Quem?”, “Onde?”, “O quê?”, e “Como?”, ou então, o problema a ser resolvido, quem irá testar, o local, o que observar e a forma dos testes, respectivamente. São inúmeras as maneiras de se responder a cada uma dessas perguntas e, a depender da combinação das respostas, se criará um formato diferente de *playtest*. Exemplos de formatos são beta (aberto ou fechado), em laboratório (Figura 3) ou terceirizado.

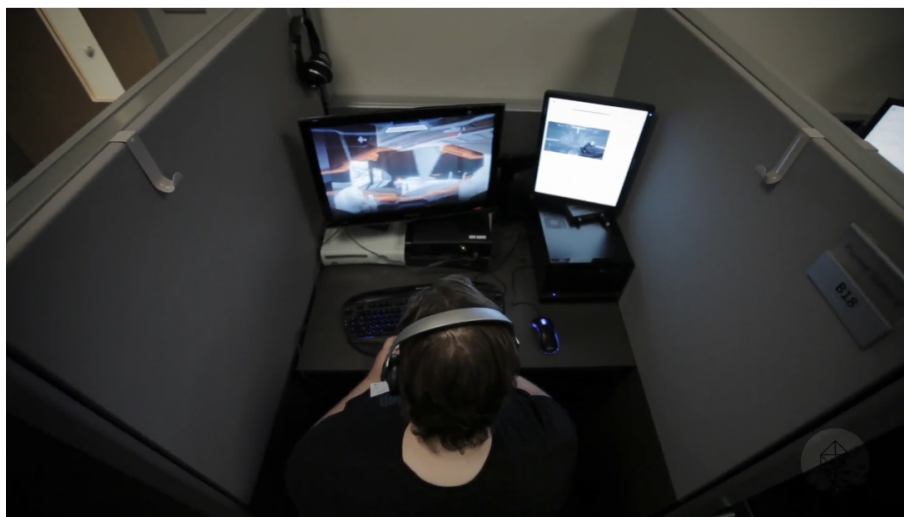


Figura 3. Cabine utilizada nos *playtests* de Halo 4. Fonte: Leone [2012]

### 3. Desenvolvimento Participativo de Jogos

Tem se tornado cada vez maior a participação dos usuários de um serviço na construção do mesmo [Sanders and Stappers, 2008], vide a criação e gestão de conteúdo de wikis,

<sup>2</sup>The Elder Scrolls V: Skyrim, RPG desenvolvido pela Bethesda, 2011.

<sup>3</sup>Exemplos extraídos do vídeo: [www.youtube.com/watch?v=K3g2840zXKY](http://www.youtube.com/watch?v=K3g2840zXKY), por WatchMojo.com.

como por exemplo a popular Wikipédia<sup>4</sup>. Na área de desenvolvimento de jogos também existe esse movimento onde o usuário (jogador) participa da criação: desde a importância dada aos *playtests*; passando pela personalização de avatares; editores de mapa; até a modificação de um jogo (*mod*) a ponto de se criar um novo, totalmente diferente [Tavares and Roque, 2007]. Um exemplo recente está no projeto do mais novo Unreal Tournament (título de renomada série) sendo desenvolvido de maneira aberta com toda a comunidade, “desde a primeira linha de código, da primeira peça de arte” [Epic Games, 2014]. As próximas subseções discutem formas de contribuição, dando alguns exemplos de valor criado pela comunidade.

### 3.1. Editores de mapa

Também conhecidos como *Map Editors* ou *Level Editors*, são ferramentas de software disponibilizadas, geralmente junto com o jogo, para a criação de novos cenários e/ou edição de mapas baseados em outros pré-existentes. Isto cria valor para os jogadores, seja pelo prazer de poder criar e experimentar, seja por facilitar a oferta de novo conteúdo para a comunidade. Alguns títulos com esta ferramenta são: Age of Empires II<sup>5</sup>, Starcraft<sup>6</sup> e Warcraft III [Blizzard, 2002].

Um grande exemplo de mapa criado pela comunidade é o Defense of the Ancients (DotA), que apesar de começar como um mapa para Warcraft III, acabou criando um novo gênero — *Multiplayer Online Battle Arena (MOBA)*. Este gênero foi definido pela criação de diversos outros títulos semelhantes, como League of Legends, Heroes of Newerth, SMITE e até o Dota 2 [Valve, 2013], lançado como um jogo completo.

### 3.2. Modificação

A Modificação ou *mod* é quando um novo jogo é criado a partir da alteração de um original. Este processo envolve modificar código e é uma tarefa complexa que requer conhecimentos técnicos específicos [Tavares and Roque, 2007].

*Mods* são difíceis de serem feitos, porém criam experiências muito diferentes do jogo original, a ponto de parecerem um jogo totalmente novo e independente. Isto se deve à maior liberdade de criação permitida pelo método, já que o criador pode alterar o código, sons, texturas e outros elementos da estética. Considerando a classificação dos elementos constituintes de um jogo [Schell, 2008]: roteiro, estética, mecânica e tecnologia, o último seria o único realmente limitado neste processo.

Half-Life é um exemplo de jogo que serviu de base para muitos *mods*, porém o mais famoso destes é certamente Counter-Strike (CS). O jogo foi desenvolvido pela comunidade e se tornou um sucesso tão grande que a Valve, criadora do título original, acabou adquirindo (sic) os desenvolvedores do CS e tornando o título um *mod* oficial [Tavares and Roque, 2007].

---

<sup>4</sup>Enciclopédia virtual. Link: [www.wikipedia.org](http://www.wikipedia.org).

<sup>5</sup>Jogo de *Real Time Strategy (RTS)* desenvolvido pela Microsoft, 1999.

<sup>6</sup>*RTS* desenvolvido pela Blizzard, 1998.

### 3.3. Criação de itens

Em jogos como Second Life<sup>7</sup>, Team Fortress 2<sup>8</sup> e Dota 2, as empresas criadoras têm um sistema elaborado para que a comunidade colabore com a criação de conteúdo, mais especificamente com a criação de itens.

No Second Life, os jogadores criam completamente o mundo virtual [Tavares and Roque, 2007]. Para tanto, existem ferramentas para criação de objetos, um sistema para sua comercialização e até uma linguagem de *script* para definir as funcionalidades de um objeto criado. Já Dota 2 e Team Fortress 2 possuem um sistema diferente, as ferramentas para criação dos itens não são fornecidas, porém a empresa disponibiliza referências de projeto e possui um sistema para avaliação dos itens que serão ou não incorporados aos jogos.

Nos três jogos citados, a criação de itens é estimulada pela possibilidade de ganhos financeiros com elas, como apresentado na Figura 4, gráfico elaborado pela Valve com o rendimento dos criadores de itens.

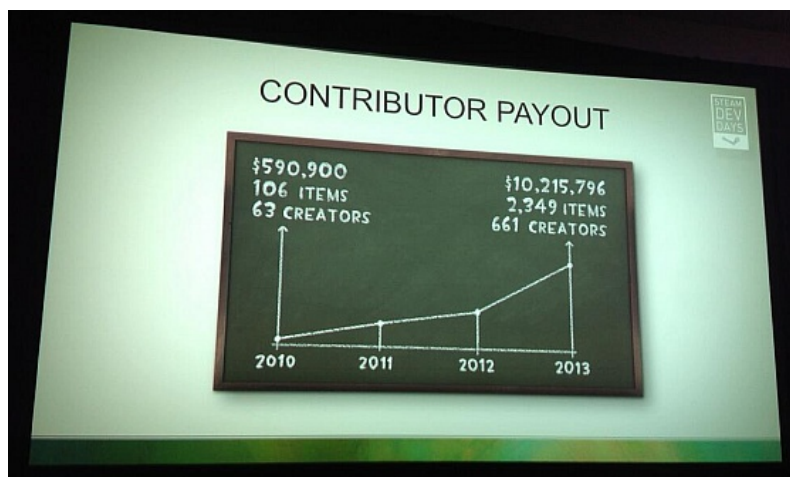


Figura 4. Rendimento dos contribuidores de Dota 2 e Team Fortress 2 de 2010 a 2013. Fonte: Hollister [2014]

### 3.4. Feedback

Foram citadas formas diretas em que os jogadores contribuem com o desenvolvimento dos jogos; o *feedback*, por sua vez, é mais indireto, onde mudanças e criações são originadas a partir da opinião dos jogadores e são postas em prática pelos *game designers*. Os *feedbacks* dos jogadores podem ser úteis desde o início do projeto, até após o lançamento deste no mercado, através de atualizações, expansões, conteúdo para download (conhecido como *Downloadable Content – DLC*) ou até um novo jogo.

Geralmente estas informações são coletadas manualmente por observação, formulários e entrevistas. Além disso, as empresas podem ter um sistema que coleta dados de interações com o jogo automaticamente [Leone, 2012], assim como podem recorrer a fóruns (tanto oficiais, quanto da comunidade), emails e outras mídias.

<sup>7</sup>Mundo virtual *online* desenvolvido pela Linden Lab, 2003.

<sup>8</sup>*First-Person Shooter (FPS)* desenvolvido pela Valve, 2007.

Um exemplo de uso do *feedback* está no corrente projeto (em estágio de desenvolvimento) da Artillery. A empresa faz dois *playtests* validando um aspecto do jogo por vez através da resposta dos jogadores [Koushik, 2014].

## 4. Modelo Proposto

Partindo da premissa de que muitos jogadores se dispõem a colaborar para o aperfeiçoamento de um jogo que eles gostem e que estas colaborações são significativas e importantes no desenvolvimento do mesmo, como discutido na seção anterior, propõe-se o modelo apresentado nesta seção.

O GamEXP é um modelo para auxiliar no balanceamento de variáveis de mecânica, automatizando diversas iterações deste processo graças a contribuição dos jogadores. O processo se diferencia da modificação, criação de itens e edição de mapas por ser mais simples e acessível. Além disso, embora seja baseado em *playtests* e *feedbacks*, estes não são aplicados da forma convencional.

As próximas subseções apresentam uma visão geral do modelo e depois o discutem em detalhe a partir de suas partes constituintes.

### 4.1. Visão geral

O modelo é constituído por quatro entidades: jogadores, jogo (cliente), servidor e o *game designer*. A Figura 5 ilustra o funcionamento do modelo desenvolvido, com as principais interações entre cada uma das entidades, formando um sistema de geração de informação para o balanceamento de configurações.

Dentre as entidades que compõem o modelo, os jogadores são a principal fonte de informação; o jogo fornece a interface de comunicação entre o jogador e o servidor; o servidor, por sua vez, coordena o experimento, com a troca, armazenamento e processamento das informações; o *game designer*, uma vez que o sistema composto pelos três primeiros está em pleno funcionamento, pode acessar as informações geradas por estes e, caso necessário, fazer alterações no servidor para adequar o processo de coleta dados.

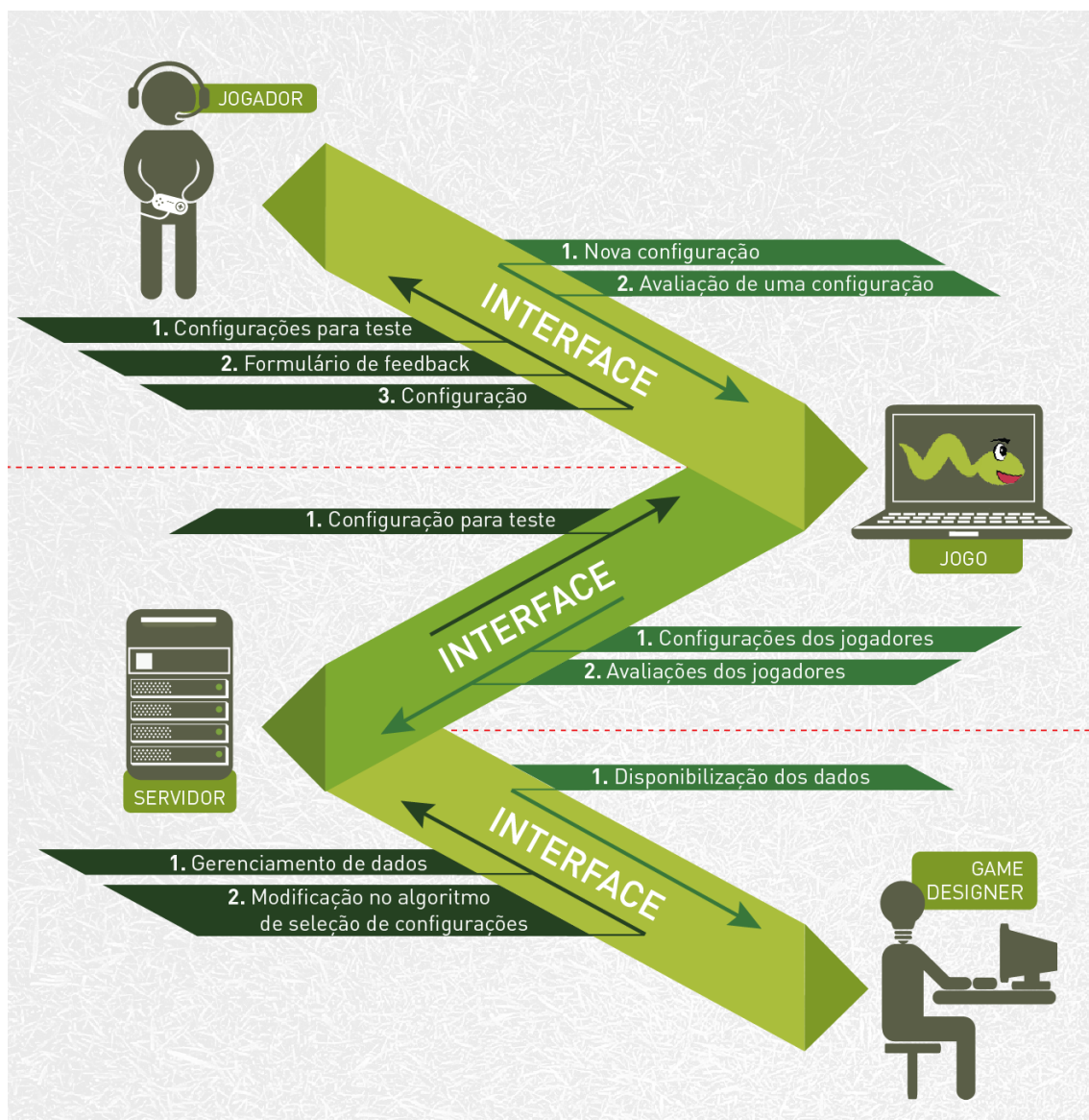
O sistema criado pela interação destas quatro entidades irá testar e gerar inúmeras configurações. Após o servidor coletar um número considerável de configurações e suas respectivas avaliações, mediante a análise dos dados coletados, o *game designer* tem informações para fundamentar a escolha de valores definitivos para as variáveis em questão.

### 4.2. Configuração

Uma configuração, neste contexto, é um conjunto composto por um valor para cada variável associada a uma regra escolhida para teste. Por exemplo, supondo um jogo simples de corrida de carros, em que a aceleração sempre assume um valor constante, analogamente o freio é uma desaceleração a um valor constante e os carros têm uma velocidade máxima. Sejam as variáveis escolhidas para teste, a aceleração, desaceleração do freio e velocidade máxima, uma configuração possível seria:

1. Aceleração =  $10m/s^2$ ;
2. Desaceleração do freio =  $15m/s^2$ ;
3. Velocidade máxima =  $200km/h$ .





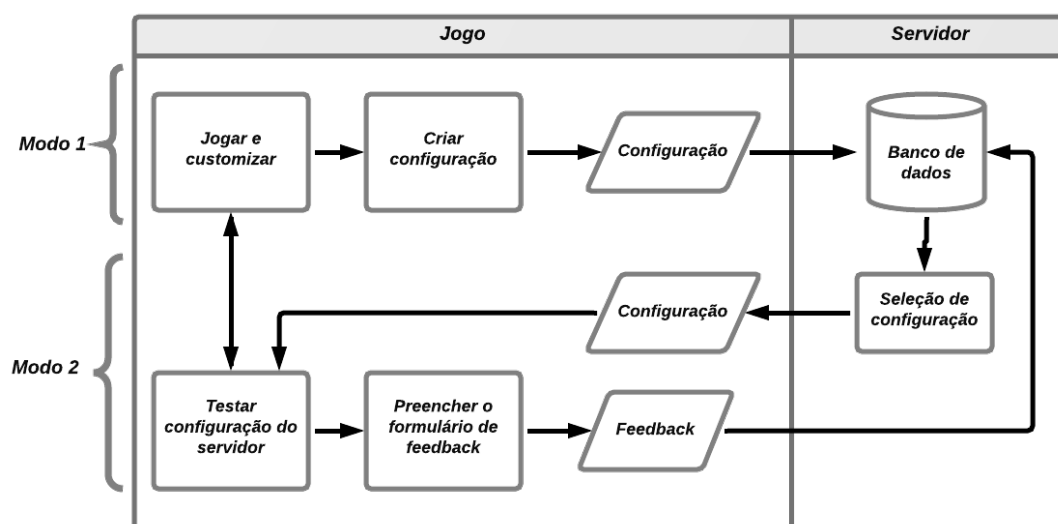
**Figura 5. Funcionamento do modelo, mostrando as interfaces entre as partes**

Esta seria somente uma das incontáveis configurações possíveis para o simples exemplo criado, que será responsável por uma experiência específica. Assim, para cada configuração diferente, decorre uma diferente jogabilidade. É um problema complexo quantificar a satisfação de determinada jogabilidade, pois não há como garantir um valor absoluto e inquestionável, pela natureza subjetiva do problema. Mesmo assim, aqui é feita uma proposta para tal quantificação dada uma configuração, onde o valor resultante (escore de satisfação) é a média aritmética de todas as avaliações (de valores quantitativos).

#### 4.3. Jogo

No contexto do experimento, o jogo é o principal objeto de estudo, ou, mais especificamente, as regras escolhidas que serão submetidos a avaliação e reconfiguração por jogadores.

Para utilizar o modelo, o primeiro passo é definir quais regras e respectivas variáveis estarão sendo testadas. Uma vez definidas, o jogo deve prover o meio de comunicar as informações criadas pelo jogador ao servidor. Para tanto, é necessário que o jogo permita ao jogador testar, criar e avaliar diversas configurações. Neste trabalho é proposto que isto seja realizado através de dois modos distintos, como mostra o fluxograma da Figura 6.



**Figura 6. Fluxograma do funcionamento do modelo**

No modo 1, o jogador pode configurar (dentro de limites pré-estabelecidos) os valores das variáveis de mecânica em questão, experimentar o jogo com a configuração criada e enviá-la para o servidor. No modo 2, ele pode testar uma configuração fornecida pelo servidor, avaliar a mesma e enviar a avaliação para o servidor.

Portanto, o jogo é considerado a interface entre o jogador e o servidor, a razão da interação e o meio pela qual ela acontece.

#### 4.4. Jogadores

Desempenham o papel de fonte dos dados no modelo. As ações responsáveis pela geração destes dados aparecem na Figura 5 e são as seguintes: teste e avaliação de configurações fornecidas pelo servidor e criação de novas configurações.

#### 4.5. Servidor

O servidor cumpre o papel de centralizador, recebendo e disponibilizando dados através de interfaces com o jogo e com o *game designer* (Figura 5).

Na interface com o jogo, o servidor dispõe de uma *Application Programming Interface (API)* para identificar um jogador, receber uma configuração e fornecer uma configuração para ser testada. Uma forma de solucionar este problema é criando um servidor HTTP<sup>9</sup>, fazendo a troca de dados no formato JSON<sup>10</sup>, porém não há nenhuma

<sup>9</sup>Hypertext Transfer Protocol, protocolo de aplicação para comunicação na World Wide Web.

<sup>10</sup>JavaScript Object Notation, um padrão de representação de dados.

restrição quanto a escolha de tecnologia.

Um ponto importante da *API*, quando o servidor retorna uma configuração para teste, é o algoritmo que irá selecioná-la. Este deve levar em conta todas as configurações presentes na base de dados. O algoritmo pode ser implementado de formas distintas, como por exemplo: escolha direta de uma configuração ou a geração de uma nova, baseada na média dos parâmetros de configurações com boas avaliações. Deve-se buscar focar os recursos de teste nas configurações mais relevantes e não investí-los nas com pouco potencial (com muitas qualificações ruins).

A interface com o *game designer* permite o acesso à base de dados. Estes dados já estarão relacionados de maneira estruturada (Figura 7), facilitando a sua consulta. A manipulação destes pode se dar por uma aplicação e/ou diretamente no acesso ao banco através de comandos SQL, ou da linguagem de manipulação de dados análoga respectiva a tecnologia utilizada na base de dados.

Desta forma, o servidor desempenha um papel semelhante ao do *game designer* no contexto de balanceamento, observando as reações dos jogadores, reconfigurando e voltando a observar as novas reações decorrentes, diversas vezes, porém em um processo automatizado que armazena estas interações de forma estruturada.

#### 4.6. Modelagem de dados

A modelagem de dados proposta se baseia em cinco entidades: jogo, jogador, configuração, *feedback* e parâmetro. Os atributos de cada entidade, assim como suas relações, estão descritos na Figura 7. A função de cada uma das entidades é apresentada a seguir.

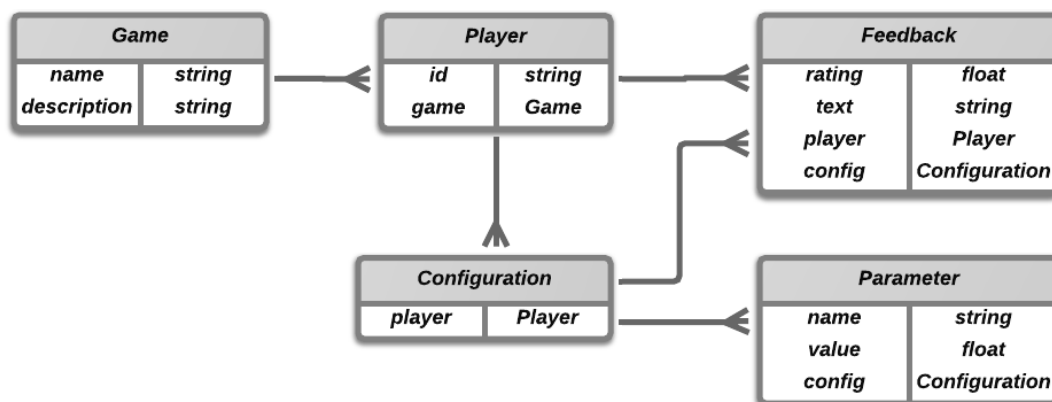


Figura 7. Diagrama entidade relacional do modelo

**Jogo (*Game*):** representa cada um dos jogos que está implementando este modelo de testes no servidor. Está no topo da hierarquia e possui zero ou mais jogadores, através dos quais se relaciona indiretamente com todas as demais entidades. Assim, os dados ficam organizados de acordo com o jogo a que pertencem, permitindo que mais de um jogo seja testado simultaneamente em um mesmo servidor.

**Jogador (*Player*):** cada jogador que conseguiu se comunicar através do jogo com o servidor. Possui uma coluna do tipo *Game* para referenciar o jogo ao qual pertence

e possui zero ou mais configurações (*Configuration*) e *feedbacks*. A coluna *id* deve identificar unicamente um jogador no seu respectivo jogo, podendo ser relacionada a um login, número de série do dispositivo que está rodando o jogo ou outro método que cumpra este papel.

**Configuração (*Configuration*):** composta por um ou mais parâmetros (*Parameter*). Uma configuração possui zero ou mais *feedbacks*. Sua única coluna, *player*, representa o jogador que a criou.

**Parâmetro (*Parameter*):** representa os parâmetros que são utilizados para formar uma configuração. Cada parâmetro possui os atributos *name*, *value* e *config*. O nome é usado para identificar a variável que receberá o valor do parâmetro; já a coluna *config* faz referência à configuração a qual pertence. Como os parâmetros podem referir-se a qualquer variável e uma configuração pode ter um número arbitrário de parâmetros, esta representação serve para qualquer configuração — desde que as variáveis assumam somente valores numéricos.

**Feedback:** Representa cada avaliação submetida por um jogador. Possui a coluna *rating*, principal métrica de avaliação, sendo um campo obrigatório; a coluna *text* possibilita a associação de uma mensagem escrita pelo jogador na avaliação (campo opcional); as demais colunas, *config* e *player*, fazem referência à configuração a que pertence e ao jogador que a criou.

#### 4.7. Game designer

O *game designer* — o termo aqui refere-se a todos os membros da equipe que participam das decisões de projeto, podendo ser um ou vários — é o principal responsável pela decisão de implementar o modelo, organizar o início do experimento, a sua finalização e, eventualmente, ajustes ao longo do caminho. Ele deverá escolher as variáveis que serão usadas; escolher quando o experimento deve terminar; e, a depender da satisfação com a coleta de dados, fazer modificações que influenciem em sua coleta, como por exemplo, alterar o algoritmo de seleção de configurações disponibilizadas para teste e/ou alterar o formulário de avaliação.

Ao fim do experimento, mediante a análise dos dados coletados, é quem irá definir os valores finais para as variáveis testadas.

### 5. Aplicação do Modelo

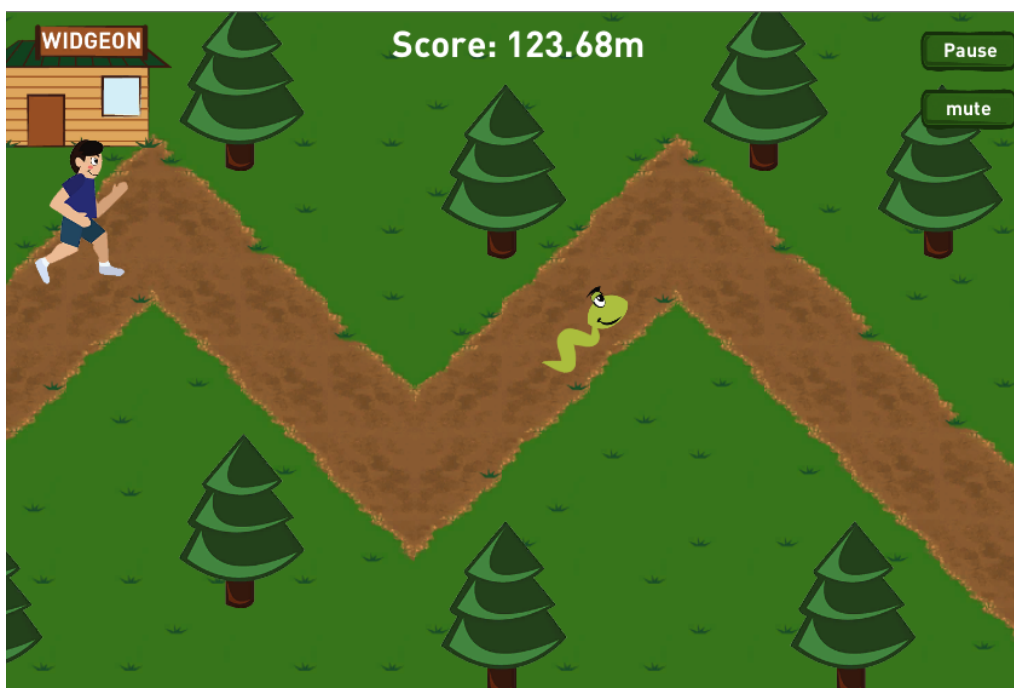
Esta seção apresenta a aplicação do modelo descrito na seção anterior. Foram implementados um jogo e um servidor para mostrar, na prática, o funcionamento do modelo desenvolvido.

#### 5.1. O Jogo desenvolvido

Para este projeto foi criado um jogo simples, do gênero *Runner*, chamado de *Chasin' Snacon* (Figura 8). O desenvolvimento foi feito na plataforma Unity3d<sup>11</sup>, usando a linguagem de programação C#. O código<sup>12</sup> está disponível em <https://github.com/arielbello/Keep-on-Track>. No Apêndice A encontra-se o *readme* do código disponibilizado com informações técnicas e de sua utilização.

<sup>11</sup>Engine com ferramentas integradas para o desenvolvimento de jogos. Site oficial: [unity3d.com](https://unity3d.com).

<sup>12</sup>O código disponibilizado é do jogo *Keep on Track!*, baseado em *Chasin' Snacon*, porém possui uma estética diferente.



**Figura 8. Captura de tela de *Chasin' Snacon*, durante uma partida**

Neste jogo o avatar é representado pela cobra verde chamada *Snacon*. O jogador controla este avatar utilizando as setas direcionais do teclado para mantê-lo na pista, enquanto ele anda a uma velocidade cada vez maior. O objetivo é ficar na pista o máximo de tempo possível, pois *Mr. Tee*, o personagem que aparece no lado esquerdo da Figura 8, está perseguindo o avatar e fica mais perto a cada momento que este sai da pista. O escore é calculado pela distância percorrida, e o jogo acaba quando *Mr. Tee* consegue alcançar *Snacon*.

Foram três as variáveis de mecânica escolhidas para os testes nesse jogo:

1. Posição inicial do avatar;
2. Velocidade inicial do avatar;
3. Aceleração do avatar.

A posição inicial do avatar se refere ao local na tela em que este irá começar, o valor determina se ele começará mais à esquerda (mais difícil) ou mais à direita (mais fácil); a velocidade inicial do avatar é, na realidade, a velocidade no começo do jogo em que todo o cenário, inclusive a pista, se move para o lado esquerdo da tela, dando a impressão de que o avatar está se movimentando para o lado direito da tela (quanto maior, mais difícil); por último, a aceleração parametriza um mecanismo do jogo que aumenta a velocidade no seu valor após um pequeno intervalo de tempo (quanto maior, mais difícil).

Com as variáveis do experimento definidas, pode-se dar como exemplo de configuração neste jogo, a que foi criada como base para os testes: posição inicial 70; velocidade inicial 40; aceleração 50.

O jogo possui dois modos, representados pelos dois botões na tela inicial (Figura 9). A opção “*Play and rate*”, que permite ao jogador testar uma configuração fornecida

pelo servidor, e “*Play and customize*”, onde o jogador pode criar configurações e testá-las (ambas de acordo com o fluxograma da Figura 6).



**Figura 9. Captura da tela inicial do jogo**

No modo de customização, na primeira vez que for usado, o jogo irá começar com a configuração padrão, a qual o jogador pode modificar a qualquer momento. Ao pausar ou acabar uma jogada, a janela da Figura 10 permite aos jogadores modificar a configuração atual e jogar novamente para experimentá-la com o botão “*Play this config.!*”; testar uma configuração do servidor, botão “*Test and Rate!*”; e para enviar a configuração criada, botão “*Submit!*” (caso ele já tenha avaliado um número mínimo de configurações).

Cada barra na Figura 10 é um controle que permite ao jogador arrastar o ponteiro horizontalmente, escolhendo um valor entre 0 e 100, que é mostrado acima da barra, após o nome do parâmetro sendo ajustado. O valor escolhido não representa diretamente um número, mas sim uma porcentagem que é usada como base de cálculo para um valor diretamente proporcional que é condizente com a regra que ele representa.

No modo de teste e avaliação o jogador tem acesso, após terminar de jogar com uma configuração carregada do servidor, à janela da Figura 11 com um formulário de *feedback* que pode ser preenchido e enviado ao servidor. O campo de texto é opcional e pode ser usado para a coleta de qualquer tipo de informação que o jogador se disponha a escrever. Abaixo do texto estão três opções mutuamente exclusivas: “*Dislike*”, “*Meh*” e “*Like*”, representando “Não gostei”, “Neutro” e “Gostei”, respectivamente; sua quantificação, seguindo a mesma ordem, é -1, 0 e 1. A escolha de uma destas opções é obrigatória para o envio do *feedback* ao servidor, uma vez que esta é a base da avaliação quantitativa da configuração relacionada.

O jogo foi desenvolvido intencionalmente para ser simples, assim como os parâmetros escolhidos foram somente três, representando regras fáceis de compreender, para

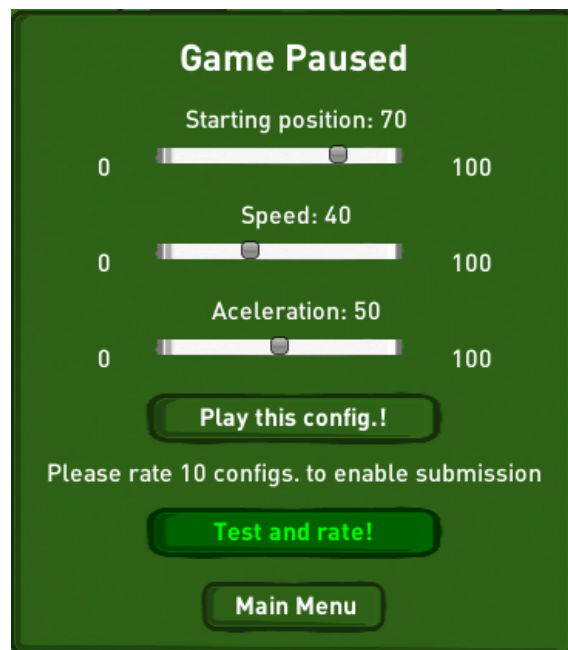


Figura 10. Captura de tela da janela de configuração

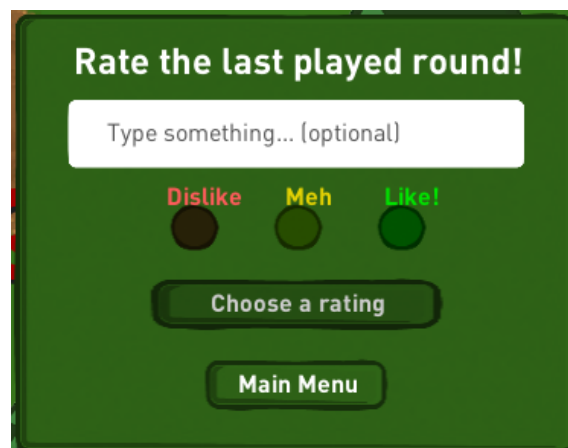


Figura 11. Captura de tela da janela de *feedback*

que os jogadores tenham consciência do impacto de cada um dos valores escolhidos ao criar uma configuração própria. Também foi tomado o cuidado para que o jogo seja rápido e muito jogável, permitindo assim que hajam muitos testes em pouco tempo, com o objetivo de ter uma amostragem significativa para um jogo que não possui muitos recursos de divulgação. Para chegar ao resultado corrente, foram feitos *playtests* assistidos ao longo do desenvolvimento do jogo, assim como coleta manual de *feedbacks*, em um processo iterativo. Isto mostra como este modelo pode ser usado de maneira complementar aos *playtests* convencionais e não possui o objetivo de substituí-los.

Como explicado acima e exemplificado nas telas do jogo, este cobre os dois requisitos principais para a aplicação do modelo, um modo de configuração e outro de testes, ambos se comunicando com o servidor quando preciso.

A versão atual está disponível online em <http://snacongame.herokuapp.com/> (fun-



ciona no próprio navegador, requer somente que o *Unity Web Player Plugin* esteja instalado no mesmo).

## 5.2. Os Jogadores

Jogadores específicos devem ser selecionados de acordo com o propósito do *playtest* a ser realizado; neste *playtest* não houve um público alvo definido, pois isto requer um tratamento especial e não é o foco do experimento. O jogo esteve disponível online para todos que tivessem acesso à internet em um sistema operacional Windows ou OS X. Em decorrência dos recursos limitados para divulgação, o número de jogadores ficou limitado, e a sua maioria é de pessoas conhecidas, por este motivo foi possível explicar brevemente o funcionamento do experimento e esclarecer eventuais dúvidas.

## 5.3. Servidor desenvolvido

Para poder testar o modelo e servir de base para futuras implementações, foi desenvolvido um servidor *web* de acordo com os requisitos, utilizando o *framework* Ruby on Rails, o banco de dados relacional PostgreSQL para persistência de dados e a plataforma de aplicações *web* Heroku para publicação. O código desta implementação está disponível em <https://github.com/arielbello/GamEXP-Server>. O Apêndice B traz o *readme* do código disponibilizado, com informações para sua utilização.

Neste servidor, o algoritmo para escolha da configuração de teste funciona da seguinte maneira: dentre todas as configurações do banco, selecionar a configuração que está há mais tempo sem receber avaliações e que não possua um *score* negativo caso tenha recebido mais de 10 *feedbacks*. Assim, todas as configurações acabam sendo testadas, visto que a prioridade é sempre para a que está há mais tempo sem receber *feedback*; além disto, as configurações que receberem 10 avaliações e tiverem uma média negativa deixam de ser consideradas, para que os recursos de testes sejam melhor aplicados em configurações com maior potencial.

## 5.4. Interface servidor – jogo

O acesso à API do servidor é ilustrado na Tabela 1. Toda a comunicação é feita usando JSON na representação dos dados.

**Tabela 1. Principais métodos da API do servidor**

Endereço relativo	Método	Descrição
/players	POST	Cria jogador
/playtest_conf	GET	Retorna uma configuração para teste
/confs	POST	Cria configuração
/feedbacks	POST	Cria feedback

O primeiro método é utilizado para fazer a identificação do jogador que está acessando o servidor. É passado por parâmetro o nome do jogo e um identificador único gerado pelo jogo, baseado no dispositivo em que está rodando. A conexão é feita caso seja criado um jogador com estes dados, ou se já existir um com o mesmo identificador no banco.



A segunda linha se refere ao método que é usado quando o jogo faz uma requisição de uma configuração a ser testada; neste momento, o algoritmo de seleção é executado e a configuração resultante é retornada.

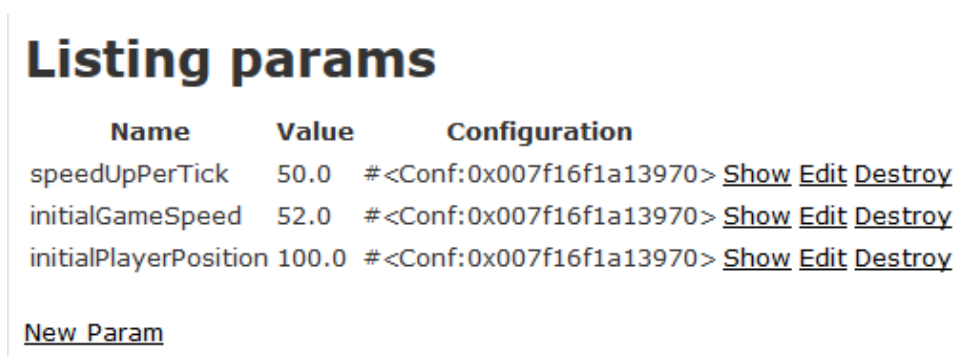
A terceira e quarta linhas representam métodos usados para o envio de configurações e *feedbacks*, respectivamente, criados pelos jogadores.

### 5.5. Interface servidor – *game designer*

O servidor implementado permite duas formas de acesso a seu banco de dados e, com isso, a todas as informações geradas no experimento.

A primeira forma é por linha de comando, já que a plataforma Heroku permite que o desenvolvedor tenha acesso ao terminal do servidor que está rodando sua aplicação. Com isto, é possível ter acesso direto ao banco, tanto pelo console Rails — que permite acesso aos dados utilizando a mesma sintaxe da aplicação —, quanto pela ferramenta de linha de comando PostgreSQL — que permite manipular os dados do banco usando SQL.

A segunda forma é visual, representando os dados do banco em conteúdo HTML. Assim, os dados também podem ser facilmente acessados e até alterados. A Figura 12 mostra um exemplo da lista de parâmetros que formam uma configuração submetida por um jogador. Esta representação tem a vantagem de possuir uma visualização mais fácil dos dados, porém tem também a desvantagem de não poder mostrar facilmente uma relação diferente — como poderia ser feito através de um comando em SQL.



Name	Value	Configuration
speedUpPerTick	50.0	#<Conf:0x007f16f1a13970> <a href="#">Show</a> <a href="#">Edit</a> <a href="#">Destroy</a>
initialGameSpeed	52.0	#<Conf:0x007f16f1a13970> <a href="#">Show</a> <a href="#">Edit</a> <a href="#">Destroy</a>
initialPlayerPosition	100.0	#<Conf:0x007f16f1a13970> <a href="#">Show</a> <a href="#">Edit</a> <a href="#">Destroy</a>

[New Param](#)

**Figura 12. Visualização HTML de parâmetros de uma configuração**

Além de HTML simples, procurou-se implementar outros meios mais intuitivos de visualização de dados, como o gráfico da Figura 13, que mostra todas as avaliações recebidas para o jogo apresentado.

## 6. Resultados e Avaliações

O experimento com o servidor e o jogo implementados durou sete dias. Durante este período o jogo e o servidor ficaram sempre disponíveis. Não foram relatados erros no funcionamento nem de indisponibilidade de serviço.

Foram 52 jogadores diferentes (de acordo com os critérios usados) gerando 156 *feedbacks* e 13 configurações. A distribuição dos valores de todas as avaliações pode ser observada no gráfico da Figura 13, mostrando que a maioria das avaliações foram positivas.

Player feedbacks

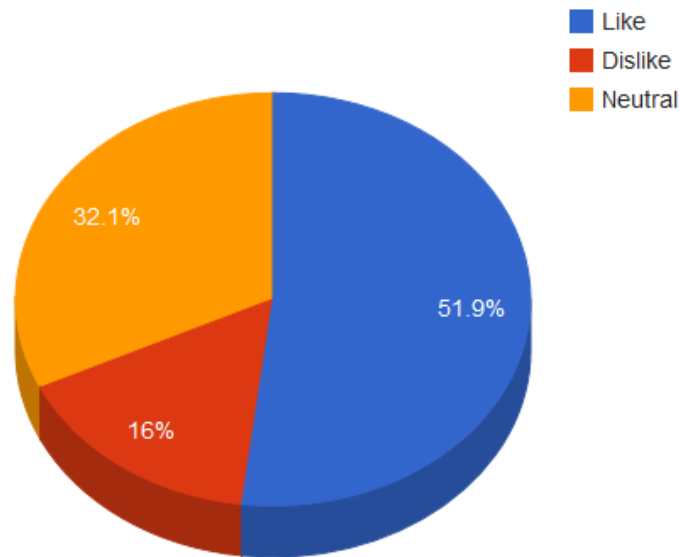


Figura 13. Gráfico mostrando todas as avaliações recebidas para *Chasin' Snaccon*

### 6.1. Avaliações quantitativas

As configurações não tiveram um número de *feedbacks* associados homogêneo, por conta do algoritmo utilizado não forçar este comportamento. Assim, as primeiras configurações a serem criadas foram as que receberam o maior número de *feedbacks*.

A Tabela 2 apresenta os resultados das avaliações das configurações, com as três melhores e as três piores, nesta ordem.

Tabela 2. As três melhores e piores configurações, conforme o resultado dos testes

Escore	Num. <i>feedbacks</i>	Param. Velocidade	Param. pos. inicial	Param aceleração
0.70	10	40	70	50
0.69	29	40	70	50
0.47	15	52	100	50
0.09	11	40	54	42
0.00	2	0	100	100
-0.20	10	0	59	0

Na tabela acima, a primeira coluna é o escore calculado de acordo com os valores -1, 0 e 1 para as avaliações “Não gostei”, “Neutro” e “Gostei”, respectivamente. Estes valores são todos somados e divididos pelo numero total de *feedbacks* recebidos, resultando em um escore no intervalo [-1, 1].

A segunda coluna apresenta o número de *feedbacks* recebidos pela configuração e as demais colunas apresentam os valores dos parâmetros para a configuração.

Os resultados apresentados mostram que a configuração criada pelo *game designer* (segunda linha), baseada em sua própria experiência, foi adequada para o jogo, com um escore alto (0.69) após 29 avaliações. Isto é confirmado também pela configuração com escore mais alto (0.70), o qual é semelhante à configuração do *game designer*, sendo que ambas possuem os mesmos parâmetros. A amostragem não foi o bastante para gerar um número de configurações e *feedbacks* grande o suficiente, com os quais se possa concluir que esta é a configuração mais apropriada para o jogo.

Apesar da pequena quantidade de dados, os valores dos parâmetros associados aos escores sugerem que uma configuração muito fácil não gera uma jogabilidade interessante, exemplificado pela menor velocidade e aceleração possíveis resultando na pior avaliação. Isto está de acordo com o conceito de balanceamento apresentado, no qual as variáveis devem estar ajustadas para criar a experiência pretendida. Neste caso, por exemplo, a intenção é apresentar um desafio de habilidade ao jogador, porém a velocidade e aceleração muito baixas não proporcionou isto ao público participante.

## 6.2. Avaliações escritas

Como apresentado na modelagem de dados do servidor (Figura 7), os *feedbacks* têm um campo para opiniões escritas (de preenchimento opcional). Neste experimento, dos 156 *feedbacks* coletados, 46 tiveram o campo de texto preenchido por 17 jogadores diferentes.

As opiniões dos jogadores foram variadas, contendo onomatopeias, elogios, sugestões e críticas de até 180 caracteres (limite adotado neste experimento). Houve algumas sugestões de novas mecânicas que deveriam ser adicionadas, críticas construtivas sobre a interface do jogo e também sobre configurações específicas (O Apêndice C traz a tabela listando todos os *feedbacks* que tiveram o campo de texto preenchido).

A associação do texto com a avaliação quantitativa, permitiu também que os jogadores expressassem mais detalhadamente do que gostaram e não gostaram em uma configuração específica. Isto ajudou no entendimento das expectativas dos jogadores e, consequentemente, no conhecimento do jogo.

Todos os textos apresentaram uma intenção de colaborar para a melhoria do jogo e, de fato, o fizeram. Pois os que foram coletados durante o beta 1.0 serviram para o desenvolvimento da versão beta 1.1 e, analogamente, deste último para o beta 1.2. Alguns exemplos de modificações feitas com base nestes textos são: possibilidade de controlar o avatar com as teclas “w” e “s” e um texto no início de cada partida mostrando os controles.

## 6.3. Questionário

Após a aplicação do modelo GamEXP, que pôde mostrar o seu funcionamento, elaborou-se um questionário (Apêndice D) com o propósito de avaliar sua relevância para um grupo potencial beneficiário de sua utilização, *game designers*.

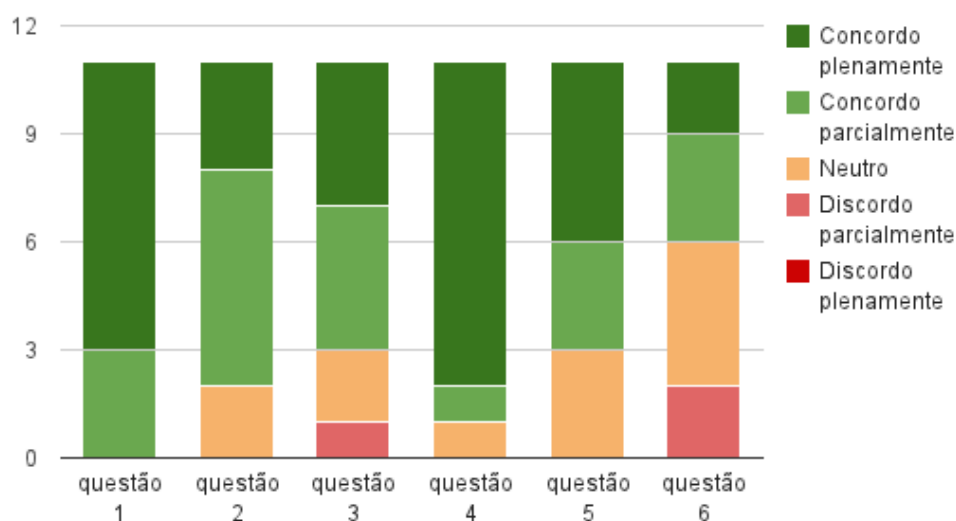
Foram 11 questionários respondidos por estudantes do curso de jogos digitais da UNISINOS, após uma apresentação do modelo, demonstração da presente aplicação e experimentação com o respectivo jogo, em um processo que durou cerca de 40 minutos ao todo; os respondentes estavam entre o terceiro e o sexto semestre de curso; todos já haviam exercido o papel de *game designer*, a maioria afirma tê-lo feito em um número entre quatro e seis jogos.

As questões referentes ao modelo foram feitas de acordo com o método proposto por Likert [1932], onde é feita uma afirmação e deve-se responder o quanto se está de acordo segundo uma escala predeterminada. As respostas possíveis eram, em ordem: discordo plenamente, discordo parcialmente, neutro, concordo parcialmente e concordo plenamente.

As afirmações feitas acerca do modelo foram:

1. Os dados gerados facilitam no balanceamento dos elementos de mecânica testados;
2. Eu aplicaria o modelo em um jogo que desenvolvi;
3. A criação de novas configurações por jogadores é um fator relevante no modelo;
4. Os dados gerados (novas configurações, *feedbacks* quantitativos e escritos) podem dar novas idéias sobre jogo;
5. O modelo permite a automação de iterações que contribuem no balanceamento;
6. Os benefícios da aplicação do modelo justificam o custo para sua implementação;

A Figura 14 mostra um gráfico com as respectivas respostas. Nota-se que as respostas foram majoritariamente positivas (“concordo parcialmente” ou “concordo plenamente”) para todas as questões, com exceção da última.



**Figura 14. Gráfico com as respostas referentes a questões do modelo**

Na primeira questão, que trata do principal resultado esperado da aplicação do modelo – facilitar o balanceamento dos elementos de mecânica testados –, todos responderam que concordam (nove plenamente e três parcialmente). As respostas da segunda mostram que mais de 80% estaria inclinado a aplicá-lo em um de seus jogos. Dos resultados da terceira e quarta questões, conclui-se que a participação dos jogadores foi entendida como importante, podendo trazer novas ideias para o jogo. A quinta questão trata de outro resultado esperado da aplicação, e teve mais de 70% de respostas positivas. A última questão listada, de custo  $\times$  benefício, teve o resultado menos expressivo, com 45% das respostas positivas, porém somente 18% de respostas negativas.

Além das afirmações citadas acima, foi feita a seguinte questão: “Conhece um trabalho que propõe algo semelhante?”. Todas as respostas referentes a esta pergunta foram “Não”, ou seja, nenhum dos questionados conhece um trabalho similar. Este resultado está de acordo com a revisão bibliográfica indicando o caráter inovador da presente proposta.

## 7. Considerações Finais

O objetivo deste trabalho foi desenvolver um modelo para o balanceamento de variáveis de mecânica com a participação dos jogadores na parametrização destes valores e avaliação da experiência decorrente de cada configuração criada. Para mostrar o funcionamento deste modelo na prática, foi implementado o experimento, utilizando o jogo *Chasin’ Snaccon* e um servidor web — ambos criados de acordo com os requisitos elicitados no modelo. Além disto foi realizado um questionário com um grupo de *game designers* acerca da relevância deste trabalho.

A realização do experimento não teve restrições de acesso para participantes, e durou sete dias. Os dados coletados serviram para entender melhor o jogo, seus jogadores, e ainda mostraram que a configuração de base proposta pelo *game designer* é satisfatória.

A interação dos jogadores com o experimento, principalmente através das opiniões escritas, reafirmou a oportunidade presente no desenvolvimento de jogos colaborativos, uma vez que cerca de 1/3 dos jogadores preencheram este campo com comentários construtivos, mesmo se tratando de um campo de preenchimento opcional.

Durante o experimento, ficou evidenciado que era preciso melhorar, no jogo, a interface pela qual os jogadores realizavam as interações necessárias para a geração dos dados. Assim, parte desta comunicação foi feita por outros meios. Este aspecto foi trabalhado fazendo uma iteração de desenvolvimento para aprimorar a interface, uma vez que esta é essencial para a aplicação do modelo.

O modelo apresentado se mostrou funcional ao longo de sua aplicação: gerenciou todos os dados de forma consistente; foi utilizado para fundamentar decisões de uma iteração de desenvolvimento do jogo criado; ajudou a entender melhor o jogador e suas interações com o jogo; e também forneceu ideias para novas mecânicas e melhorias para o jogo. Em trabalhos futuros, será aplicado a jogos de maior complexidade para complementar os resultados aqui alcançados.

No questionário de avaliação realizado com um grupo de *game designers*, todos concordaram com a utilidade da aplicação do modelo na geração de dados para balanceamento de variáveis, além de afirmarem não conhecer nenhum trabalho semelhante a este.

Sendo assim, a aplicação do modelo mostrou-se útil, pois os resultados indicam que, ao dispor de um volume maior de dados, este pode responder de maneira convincente à questão de balanceamento de variáveis de mecânica. Os dados obtidos ajudam a entender configurações muito ruins e outras potencialmente satisfatórias; *feedbacks* dos jogadores sinalizam pontos fortes do jogo, pontos a serem aprimorados e servem também como fonte para ideias novas a serem incorporadas no mesmo. Além disto, há uma percepção positiva de *game designers* sobre a utilização do modelo.

Ficam como contribuições deste trabalho para a área de desenvolvimento de jo-

gos: o modelo desenvolvido, um exemplo de sua implementação e a disponibilização dos códigos do servidor e do jogo. Por fim, esta pesquisa exploratória dispõe resultados que mostram uma nova perspectiva que soma no caminho promissor do desenvolvimento de jogos em conjunto com os jogadores.

## Referências

- Adams, E. and Dormans, J. (2012). *Game Mechanics: Advanced Game Design*. New Rider Games, 1st edition.
- Blizzard (2002). Warcraft 3: Reign of chaos. *RTS*. Disponível para Windows e OS X.
- Blizzard (2010). Starcraft 2 series. *RTS*. Disponível para Windows e OS X.
- Davis, J. P., Steury, K., and Pagulayan, R. (2005). A survey method for assessing perceptions of a game: The consumer playtest in game design. *The International Journal of Computer Game Research*, 5(1).
- Eletronic Arts (2009). The Sims 3. Simulação social. Disponível para Windows e OS X.
- Epic Games (2014). Unreal Tournament Development. Disponível em: [https://wiki.unrealengine.com/Unreal\\_Tournament\\_Development](https://wiki.unrealengine.com/Unreal_Tournament_Development). Acesso em: 5 de novembro de 2014.
- Fullerton, T. (2008). *Game Design Workshop: A Playcentric Approach to Creating Innovative Games*. Elsevier Inc, 2nd edition.
- Hollister, S. (2014). On average, ‘team fortress 2’ and ‘dota 2’ item creators made \$15,000 last year. Disponível em: <http://www.theverge.com/gaming/2014/1/16/5316248/on-average-team-fortress-2-and-dota-2-item-creators-made-15000-last>. Acesso em: 6 de novembro de 2014.
- Interplay (1997). Fallout. *RPG*. Disponível para MS-DOS, Windows e Mac OS.
- Koushik, M. (2014). Building Games Through Playtesting. Disponível em: <http://blog.artillery.com/2014/06/building-games-through-playtesting.html>. Acesso em: 6 de novembro de 2014.
- Leone, M. (2012). Data entry, risk management and tacos: Inside halo 4’s playtest labs. Disponível em: <http://www.polygon.com/2012/10/24/3538296/data-entry-risk-management-and-tacos-inside-halo-4s-playtest-labs>. Acesso em: 1 de novembro de 2014.
- Likert, R. (1932). A technique for the measurement of attitudes. *Archives of Psychology*, 140.
- Nintendo (1990). Super Mario World. *Platformer*. Disponível para SNES.
- Salen, K. and Zimmerman, E. (2004). *Rules of Play: Game Design Fundamentals*. MIT Press, 1st edition.
- Sanders, E. B. N. and Stappers, J. (2008). Co-creation and the new landscapes of design. *CoDesign: International Journal of CoCreation in Design and the Arts*, 4(1).
- Schell, J. (2008). *The Art of Game Design: A Book of Lenses*. CRC Press, 1st edition.
- Tavares, J. P. and Roque, L. (2007). Games 2.0: Participatory Game Creation. Disponível em: <http://projeto.unisinos.br/sbgames/anais/gameecultura/shortpapers/34757.1short.pdf>. Acesso em: 2 de novembro de 2014.
- Valve (2013). Dota 2. *MOBA*. Disponível para Windows, OS X e Linux.

## A. *Readme* do Código do Jogo Disponibilizado

Keep on Track!

=====

This is a 2D runner game made to implement the GamEXP model. It was made with Unity3d free version, using C# and some third party libraries (listed below).

### Whats GamEXP

---

You might have this question. Basically, it's a way to generate statistics based on which you may balance your game mechanics. This model relies on players to create new configurations to the game as well as rate them. The data is processed and collected by a server. You can find an example code of such a server here: <http://github.com/arielfbello>.

So, to implement the GamEXP you'll need: - Players to test, tweak and evaluate - A server to handle the data - A game to experiment with

### Basic Setup

---

To quickly setup the GamEXP with this game and my server example, you need to first setup the server (can be done locally). Then you should edit the Constants.cs (found on this game Assets/Scripts folder), typing your server url and the game name you've set up for this experiment on the server.

Last but not least, take your time reading and understanding the code (it's not good, I guarantee it). If you need some support, mail me: [arielfbello@gmail.com](mailto:arielfbello@gmail.com)

### Credits

---

First, I'd like to thank Matt Schoen for the great JSONObject library I use here. Link (Unity asset store): <https://www.assetstore.unity3d.com/en/#!/content/710>

Second, Thanks to JNA Mobile, who put together the really nice GUI I try to use in this game. Link (Unity asset store): <https://www.assetstore.unity3d.com/en/#!/content/7987>

As for my code, it's using the beer license as follows

THE BEER-WARE LICENSE (Revision 42): <[arielfbello@gmail.com](mailto:arielfbello@gmail.com)> wrote this file. As long as you retain this notice you can do whatever you want with this stuff. If we meet some day, and you think this stuff is worth it, you can buy me a beer in return



## **B. Readme do Código do Servidor Disponibilizado**

### GamEXP Server

=====

This is a Rails app developed to implement the GamEXP model.

### Whats GamEXP?

You might have this question. Basically, it's a way to generate statistics based on which you may balance your game mechanics. Take a look at db/schema.rb for how this data is represented.

This model relies on players to create new configurations to the game as well as rate them. The data is processed and collected by a server (such as this one here).

So, to implement the GamEXP you'll need: - Players to test, tweak and evaluate;  
- A server to handle the data; - A game to experiment with.

### Basic Setup

To quickly setup the GamEXP, you should setup this server and a game to work with (here's one that does just that: <https://github.com/arielfbello/Keep-on-Track>).

To setup the server you need to: - Configure its secrets on config/secrets.yml;  
- Set the db credentials on config/database.yml (The database used on this example is PostgreSQL, but you can use your preferred database adapter) - Create your database and run rake db:migrate - Setup the db/seeds.rb with your base data. Here you need to create at least the game. Pay attention to the game's name, as you will need to setup the same on the client (the game itself).

With the server up and the game hooked up with it, the experiment can begin!

Last but not least, take your time reading and understanding the code (it's not good, I guarantee it). If you need some support, mail me: [arielfbello@gmail.com](mailto:arielfbello@gmail.com)

### Credits

###THE BEER-WARE LICENSE (Revision 42): <[arielfbello@gmail.com](mailto:arielfbello@gmail.com)> wrote this file. As long as you retain this notice you can do whatever you want with this stuff. If we meet some day, and you think this stuff is worth it, you can buy me a beer in return

### C. Todos os *Feedbacks* com Texto Coletados

Rating	Text	player id
1	Precisa dar mais tempinho pra cobrinha correr, antes do gordinho pegá-la.	3
1	Curti muito! é viciante! gostei da expressão da cobra que indica pra a gente se as coisas estão indo bem ou não.	8
-1	i think too fast acceleration	11
0	bota um boost	9
1	Awesome!!!!	12
0	Maybe or up and down or w and s	12
0	Fastest game over with this config	12
-1	Snake lerda!	12
0	Mais coisas pra cobrinha se esconder, po... qualquer buraco entre as árvores ela é pega tadinha!!	12
-1	zzzzZZZiZzz	12
1	If you leave the track for any kind of reason, you start to get closer to the guy and I can't see any chance to get far again. Maybe a speed modifier every x seconds on the track lasting for few secon	14
-1	Para que essa cobra???	19
-1	??????????	19
-1	A gente é a cobra??	19
-1	A cobra quer fugir do cara?	19
0	Entendi agora...	19
0	Essa foi mais difícil	19
0	Pq vc n coloca a Nebuchadnezzar tentando fugir dos sentinelas pela linha mecânica tentando chegar em Zion?	19
0	Dá dando para se divertir	19
-1	Mto lenta essa config, comecei a filosofar sobre a vida enqto jogava até q n aguentei e me suicidei (no jogo)	19
0	Mais emocionante	19
0	Legal essa	19
1	I think it could be a little bit faster starting out. Looks awesome though! You rock Ariel! ;D	21

-1	Acho que era para avisar como jogar antes do jogo iniciar, dar um jeito da pessoa saber que esta controlando a cobra a primeira vez achei que eu era o cara.	12
1	Essa configuracao mais rapida é a melhor	12
0	Acho que deveria ter uma opcao mutiplayer, o perseguidor no W;S e a cobra no direcional.	12
1	I think the rate of speed of the player and the starting distance between the player and the runner is great. When the games starts freeze the game to allow the player to ready up.	25
1	When the game starts place the player on a long upward slant so they can get momentum going then give them a small stretch of close slants. Also periodic speed boots for the player wouldn't hurt.	25
1	I enjoy this game...i think the road needs to start off a little bit easier to grab the gamer and then get harder	27
0	I think that you could put a little intro on how the game works... when playing the first time... or something like that.. otherwise it a ok game... continue the good work	29
-1	Type something... (optional	30
0	More variety would help hold my attention for longer. Good starting idea!	37
0	Nothing slows down the man, but too many things slow down the snake! There's nothing to help/get back at previous place if the player messes up	37
-1	not far enough ahead!	11
-1	Accelerated too fast	11
0	I like but a acceleration too fast	11
-1	too fast acceleration	11
-1	too far forward to start	11
1	Cool!	38
-1	Too slow, a bit boring.	38
0	The speed was nice, but it started too close to the guy.	38
1	Fun!	38
1	Nice, but there was a long part with only short zigzags.	38
1	Needs more directions and objectives	45
1	Very engaging! Love the concept, audio, and animation!	50
0	Some strange lags...	49

## D. Questionário aplicado

### Questionário GamEXP

Este questionário visa coletar percepções de game designers a respeito da relevância do modelo GamEXP, uma vez que são usuários em potencial. Os dados são coletados preservando a anonimidade do sujeito. As respostas serão utilizadas no trabalho final (um artigo estendido) referente a este projeto.

\* Required

**1. Quantos jogos você já criou exercendo a função de game designer? \***

considerar também situações em que exerceu esta função parcialmente, por exemplo: participava em decisões de game design, mas era o principal programador.

- ☐ Nenhum
- ☐ 1
- ☐ de 2 a 3
- ☐ de 4 a 6
- ☐ de 7 a 10
- ☐ mais de 10

**2. Você é estudante de graduação em jogos digitais ou curso similar? \***

- ☐ Não
- ☐ Sim

**3. Caso tenha respondido sim na pergunta anterior, em que semestre está?**

- ☐ primeiro
- ☐ segundo
- ☐ terceiro
- ☐ quarto
- ☐ quinto
- ☐ sexto
- ☐ Other:

As questões a seguir são referentes ao trabalho de título GamEXP apresentado. As questões 4 a 10 apresentam afirmações. Responda cada uma selecionando uma única alternativa que melhor descreva o grau em que você discorda ou concorda com a afirmação.

☐ Ok!

4. Os dados gerados facilitam no balanceamento dos elementos de mecânica testados. \*

Discordo plenamente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo plenamente
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

5. Eu aplicaria o modelo em um jogo que desenvolvi \*

Discordo plenamente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo plenamente
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

6. A criação de novas configurações por jogadores é um fator relevante no modelo \*

Discordo plenamente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo plenamente
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

7. Os dados gerados (novas configurações, feedbacks quantitativos e escritos) podem dar novas idéias sobre jogo \*

Discordo plenamente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo plenamente
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

8. O modelo permite a automação de iterações que contribuem no balanceamento \*

Uma iteração é considerada um ciclo de (em ordem): protótipo, teste, análise e refinamento.

Discordo plenamente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo plenamente
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

9. Os benefícios da aplicação do modelo justificam o custo para sua implementação \*

Discordo plenamente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo plenamente
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

10. Conhece um trabalho que propõe algo semelhante? \*

- ☐ Sim
- ☐ Não

11. Caso tenha respondido que sim, dê referências deste trabalho (nome, onde encontrar, autor etc)

12. Críticas e sugestões

Sinta-se à vontade para deixar críticas ou sugestões sobre o projeto.

Submit

Never submit passwords through Google Forms.

100%: You made it.