Planeta Azul: A água

Carlos Leite and José Ornelas and Mohamed Ali
Faculdade de Ciências da Universidade do Porto

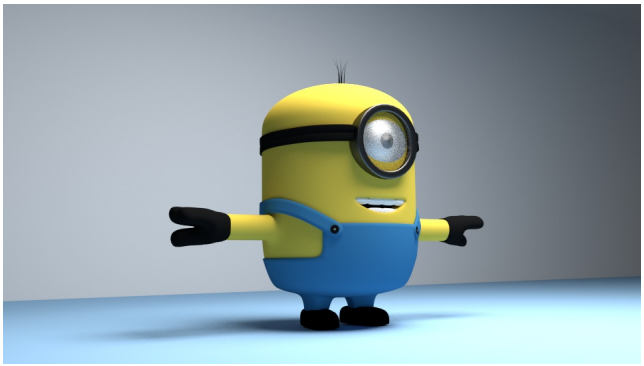
Este projeto é um jogo 2D, de plataformas, elaborado em Unity. O objetivo do jogo é alertar a população sobre a escassez de água potável e introduzir novos comportamentos que conduzam a um consumo mais moderado da mesma.

Gameplay: ⟨http://youtu.be/HYdJ2tzYcyQ⟩.

Categories and Subject Descriptors: [**Sistemas Multimédia**]: Projeto Final—*Videojogos*

General Terms: Sistemas Multimédia, Água, Videojogos

Additional Key Words and Phrases: Sustentabilidade, Hábitos, Unity, RpgMaker, Jogo 2D



1. INTRODUÇÃO

O corpo humano é constituído por 70% de água, esta não pode ser reproduzida nem substituída tornado-se portanto num bem essencial de primeira necessidade. Assim sendo, a água deverá ser preservada e salvaguardada. No entanto, devido ao consumo excessivo de água e à poluição, cerca de 1.2 biliões de pessoas não têm acesso a água potável. Prevê-se que em 2027 dois terços da população mundial irá enfrentar escassez de água potável [FAO 2013]. O problema é que grande parte da população não está a par destes dados.
Assim sendo, desenvolvemos um jogo 2D de plataformas, utilizando o Unity, onde o objetivo é alertar para a escassez de água potável e introduzir novos hábitos de consumo.

2. STATE OF THE ART

2.1 Hábito

O ser humano é definido por comportamentos e hábitos segundo a teoria cognitiva comportamental da psicologia[Stanford Encyclopedia of Philosophy 2000]. Esses hábitos e comportamentos podem ser alterados, apagados e reformulados.
Um hábito é uma ação realizada em piloto automático, ou seja, uma ação não racionalizada, gravada no subconsciente. Forma-se a partir de um comportamento repetido várias vezes, durante 21 dias [Dean 2013]. Cerca de quarenta por cento das ações que realizamos diariamente são baseadas em hábitos [Duhigg 2014].

2.2 Comportamento

Utilizamos o modelo de Fogg [Dr. BJ Fogg, Stanford 2011] para definir um comportamento. Um comportamento é definido por três fatores: motivation + trigger + ability (Figura 1).
Motivation é o desejo de realizar o comportamento. Trigger é aquilo que despoleta o comportamento. Ability é o que indica se existem meios para prosseguir com o comportamento.
Quando um destes fatores falha então, por consequência, não se dá o comportamento esperado.

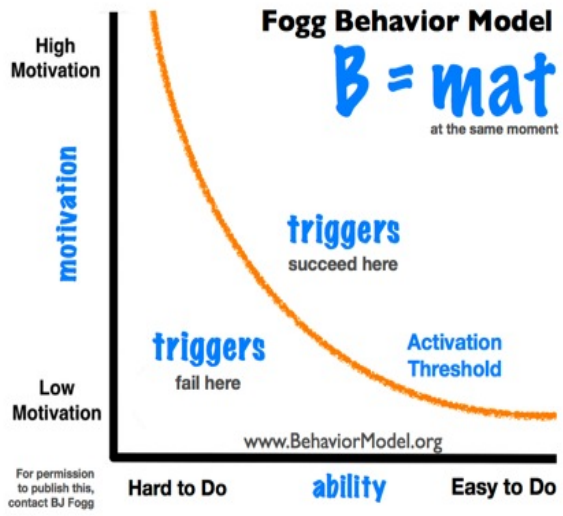


Fig. 1. Fogg Model.

Adaptamos o modelo de Fogg ao jogo que desenvolvemos. Para um comportamento como, por exemplo, fechar a água da torneira temos que: fechar a água da torneira = acabar o nível no menor tempo possível (motivação) + um relógio indicando quantos segundos faltam para o final do nível (trigger) + acesso a um computador (ability).
Outros comportamentos implementados podem ser encontrados em [Construção Sustentável 2013].

2.3 Paralelismo com os videojogos

Nesta secção iremos explorar o que é necessário para criar o hábito de jogar um videojogo e como é que o jogador consegue aplicar os comportamentos que tem no jogo à vida real.

2.3.1 *Mecânicas de jogo.* Existem um conjunto de técnicas utilizadas pelos designers de jogos que são responsáveis para entreter o jogador, [Schell 2008], e fazer com que ele volte a jogar esse jogo. Essas técnicas têm o nome de mecânicas de jogo. São elas que nos levam a resolver puzzles, a querer destruir inimigos e a explodir blocos.
No desenvolvimento do jogo basemo-nos na flor fundamental, Figura 2, [Schell 2008].

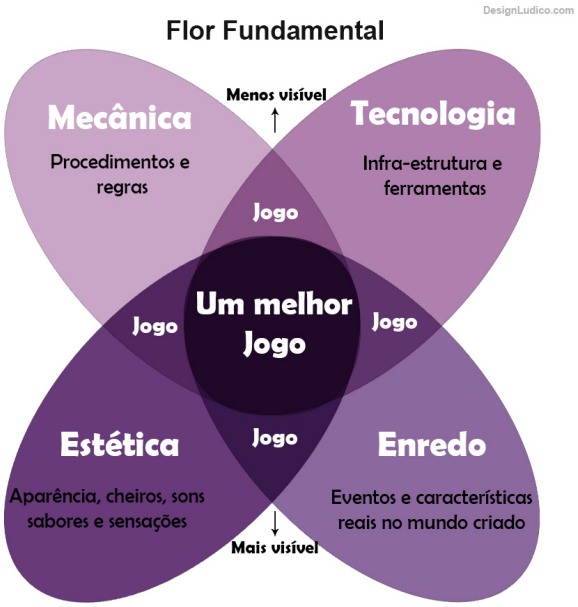


Fig. 2. Flor fundamental.

Nas secções de game design e game development serão exploradas as mecânicas de jogo utlizadas.

2.3.2 *Aplicação dos comportamentos.* Através de boas mecânicas de jogo é possível tornar a ação de jogar num hábito [Lauren Hall-Stigerts 2013]. Através desse hábito será possível reforçar e introduzir comportamentos, referidos na secção anterior, que o jogador irá aplicar no seu quotidiano [David Williamson Shaffer et. al 2004].
Os jogadores conseguem usar aquilo que aprendem nos jogos em várias situações. Esse tipo de aprendizagem tem a designação de Situated learning [John SeelyBrown et. al 1989].
Jogos muito violentos, como Grand Theft Auto, [Rockstar 1998], já foram banidos em alguns países, pois, alguns jogadores replicaram comportamentos violentos do jogo na vida real. Outras referências para estudos que suportam o uso de jogos para aprendizagem e introdução de comportamentos, ver [David Williamson Shaffer et. al 2004].
Diversas aplicações e jogos, como o HabitRpg, [Habbit RPG 2012], utilizam a metologia aqui descrita para introduzir hábitos através de videojogos.

2.4 Categoria do jogo

2.4.1 *2D Plataformas.* Elaboramos um jogo 2D pois este apresenta uma jogabilidade simples e a sua implementação é de menor dificuldade em relação a um jogo 3D, uma vez que a câmara apenas se movimenta em dois eixos, tal como o player.
Assim sendo, segundo a flor fundamental, Figura 2, obtemos uma mecânica mais simples com menos variáveis de decisão referentes à jogabilidade. Uma mecânica simples é um marco a atingir na elaboração de um jogo [John Rose 2008].
A escolha do estilo de jogo, plataformas, deve-se à jogabilidade e às mecânicas simples que este requer. Através de um jogo de plataformas é possível implementar os conceitos sobre comportamentos e hábitos referidos nas secções anteriores.
Existem vários jogos de plataformas, 2D, com sucesso, [Super Mario, Shigeru Miyamoto 1985].
No entanto, enquanto que o comportamento do jogador no Super Mario é partir blocos com a cabeça e entrar dentro de canos, neste projeto, um comportamento será, por exemplo, desligar torneiras.

2.4.2 *Ferramentas.* Atualmente existem várias ferramentas para criação de jogos. O GameMaker da YOYOGAMES [GameMaker 2013] possibilita a criação de jogos sem conhecimento de programação. Como ponto forte destaca-se a comunidade online onde são disponibilizados tutoriais de como criar jogos 2D e 3D [GameMaker Community 2013]. No entanto a versão gratuita é limitada. Não permite importação de objetos 3D, manipulação de câmaras, não permite implementar uma jogabilidade fluida nem manipulação de texturas.
O Unity, [Unity Technologies], é um game engine que permite a criação de jogos 2D e 3D multi plataforma. Ainda na versão free permite programar nos shaders, importar objetos de outros softwares, como 3ds Max, Maya e Blender e permite criar scripts em C, Javascript e Boo.
O Unity também dispõe de uma comunidade online com tutoriais que abrangem o desenvolvimento de um jogo 2D e 3D [Unity Community]. Por essas razões o Unity foi o game engine escolhido para desenvolvimento do jogo.

3. GAME DESIGN

O jogo é constituído por um nível onde a 'scene' é uma casa. O personagem pode realizar acções pela 'scene' tais como caminhar, saltar, reparar e desligar objectos.
Existem cinco tarefas, desligar cinco torneiras. Ao concretizar as cinco tarefas o jogo acaba e o jogador receberá uma pontuação consoante o tempo em que conseguiu realizar as tarefas. Se não conseguir realizar as tarefas num tempo definido para o nível então o jogador terá de recomeçar nesse nível.
A pontuação irá ser usada para lhe atribuir uma medalha que pode ser de Ouro, Prata ou Bronze.
A pontuação e o desejo de transitar para o nível seguinte, com uma dificuldade acrescentada, serão os rewards do player. Este sistema de reward foi baseado em jogos como Angry Birds e Cut the Rope.

3.1 Menu

Os menus foram criados no Unity utilizando o UnityGUI, pois permite a criação de menus de forma rápida e funcional [UnityGUI 2012].
Definimos dois menus. Um deles é mostrado quando se inicia o jogo, menu principal, Figura 3. Podemos através dele começar um novo jogo ou continuar um jogo gravado anteriormente e ver as instruções de como jogar.
O menu principal contém uma imagem do jogo de modo a suscitar interesse no utilizador [Amanda Sibley 2012]. O segundo menu, Figura 4, trata-se do menu de pausa que é mostrado quando o utilizador pressiona a tecla Esc. Usamos a tecla Esc para pausar, pois, utilizamos o conhecimento que o jogador tem de outros jogos e aplicações (onde essa tecla é utilizada para aceder a esse tipo de menu). Dentro deste menu temos a opção para resumir o jogo, aceder às opções e sair da aplicação.

Fig. 3. Menu principal

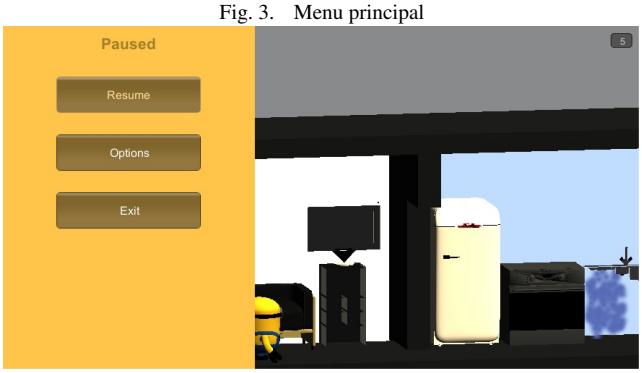


Fig. 4. Menu de pausa

3.2 Character

Na criação do nosso character foi utilizado o Blender, Figura 5, uma vez que este é gratuito e conta com uma larga e ativa comunidade online com tutoriais disponíveis [Blender Guru 2014]. O Blender permite uma modelação dos objetos rápida, devido aos diversos atalhos que possui, ao edge slide, collapse, dissolve e à utilização de scripts em Python. O Blender também possui um esqueleto (Rigg) na forma de um humano o que é um fator importante para o Rigging do nosso character e tem uma toolset completa para animação, por exemplo, walk-cycles automáticos [Blender Features 2014].

Para o nosso character escolhemos um Minion do Gru [Pierre Coffin, Chris Renaud 2010], pois está associado a um filme de sucesso. É um character conhecido globalmente e dessa forma é responsável por parte da curiosidade do jogador em testar o jogo [Joshua Porter 2013].
Para a ciação do Minion foi seguido um tutorial [Blender for noobs 2013].
Para o processo de rigging utilizamos uma armadura humana, uma human meta rigg em que tivemos que modificar a estrutura, pois, o Minion não tem todo o corpo à escala de um humano. Também tivemos de apagar ossos, pois este apenas tem três dedos.
Na animação utilizamos o pose editor onde criamos frames através control bones para animar de forma realista as pernas e braços do Minion.

Fig. 5. Character no Blender

3.3 Scene

A casa foi elaborada no Blender por intermédio das ferramentas line e box.
Utilizaram-se diferentes proporções nas box. Com isto criaram-se as plataformas de pavimento e cobertura. Posteriormente utilizaram-se box's no plano vertical para criar as divisões entre as áreas da casa: cozinha, lavandaria, elevador, sala, quarto (Figura 6), garagem e casa de banho. Na criação das portas utilizamos a ferramenta (Proboolean -?-). Esta permite-nos remover pequenos pedaços do plano vertical e obter o efeito da porta. Dessa forma o character pode andar pela casa toda.
Para que cada espaço tivesse uma textura diferente e se tornasse mais característico e identificável, foram criados planos verticais (plan) que se encontram no fundo de cada área da casa. Desta forma é possível dotar cada espaço de diferentes materiais. Utilizou-se para isso o menu "Materials" de onde se escolheram os materiais a aplicar.
No recheio da casa utilizamos modelos 3D, como por exemplo o frigorífico, o carro e a cama, [Archive3D 2012], para que as divisões da casa fossem facilmente identificadas.

3.4 Illumination

Para a iluminação da casa foi utilizada apenas uma direction light, pois esta ilumina de forma uniforme toda a cena.

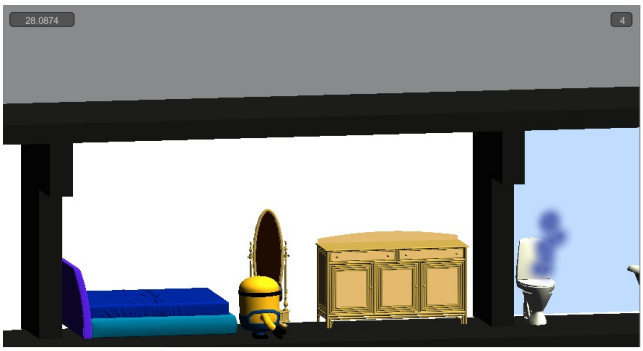


Fig. 6. Quarto

4. GAME DEVELOPMENT

4.1 Sistemas de partículas

Um sistema de partículas é uma técnica de Computação Gráfica que gera um número (definido) de partículas. Através da ferramenta Particle System do Unity podemos criar partículas que simulem a água.
Os sistemas de partículas podem ser emitidos em formas geométricas, tais como cones ou paralelepípedos. As formas dos pontos de emissão permitem criar diversos efeitos. A água a cair da torneira, por exemplo, utiliza um cone com um ângulo reduzido, enquanto que a água a saltar da sanita ou máquina de lavar utilizam um cone com um ângulo um pouco maior. Isto permite fazer o efeito de água a saltar para cima.
No caso da banheira e da banca definimos o limite do sistema de partículas como um paralelípedo. Isso permite criar o efeito de cortina de água a escorrer, Figura 7.

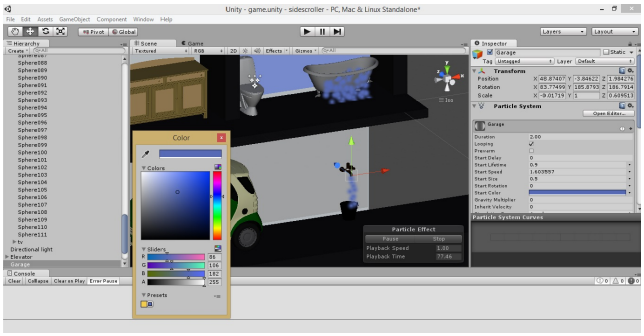


Fig. 7. Sistema de partículas

4.2 Deteção de colisões

A detecção de colisões é usada para o character poder percorrer a Scene. No Unity 3D esta é feita através de Colliders [Unity documentation 2013]. No nosso character utilizamos uma box collider que envolve todo o boneco. No caso do cenário este também precisa desta componente, contudo visto que envolve formas mais complexas utilizamos um Mesh Collider [Unity documentation 2012a], isto permite que toda a Mesh da scene seja coberta pelo sistema de colisões.
Para a deteção da colisão entre o character e os objetos foram utilizados Ray Casts [Unity documentation 2012b] de modo a calcular a distância entre os objetos.

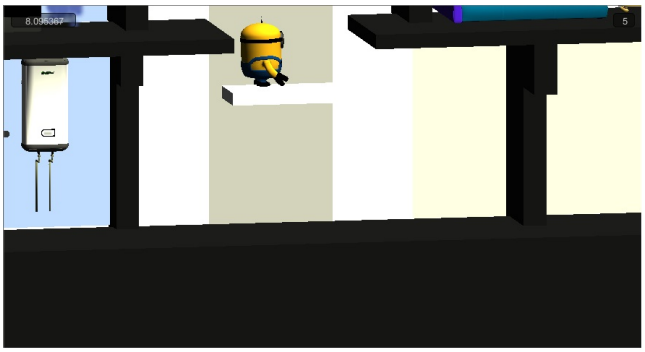


Fig. 8. Elevador

4.2.1 *Elevador.* O primeiro nível do jogo tem 2 pisos. Através do elevador, Figura 8, é possível que o character se movimente entre o Piso 0 ou o Piso 1 da casa. Pode ser accionado e chamado utilizando a tecla E.
A plataforma do elevador é feita através de um objeto do tipo cube, e por isso o sistema de colisões do elevador é realizado através do cube collider. No entanto, a plataforma do elevador, tem uma particularidade em relação às restantes plataformas, pois é a única que se movimenta na cena. Dessa forma foi necessário implementar um sistema que conjuga-se a deteção da colisão character-plataforma e que simultaneamente atualiza-se a posição vertical do character.

5. ACKNOWLEDGEMENTS

A realização deste trabalho dependeu bastante dos conhecimentos que conseguimos obter sobre o Blender e o Unity. Ambos possuem comunidades online muito boas.
A Blender User Community compila uma panóplia de sites onde a comunidade oferece gratuitamente ajuda, tutoriais e cursos de treino.
Na Unity Community oferece gratuitamente ajuda, tutoriais, scripts, assets, entre muitas outras coisas.
O facto destas comunidades serem tão grandes a nível de utilizadores e tão ricas em conteúdos foi preponderante para a realização e conclusão do nosso jogo.

6. RESULTADOS E CONCLUSÃO

Obtemos um demo jogável exportável para qualquer plataforma. A demo é constituída por um nível, Figura 9.
Através deste projeto tivemos a oportunidade de adquirir conhecimentos em áreas que se encontram fora do escopo do plano de estudos do curso, como, por exemplo a psicologia comportamental, e aprofundar conhecimentos na área da computação gráfica.
Desenvolvemos conhecimentos na criação de objetos utilizando o blender, Figura 5, e implementamos conceitos de computação gráfica utilizando o Unity: deteção de colisões, Figura 8, sistemas de partículas, Figura 10, luzes e câmaras.

REFERENCES

Amanda Sibley. 2012. 19 Reasons You Should Include Visual Content in Your Marketing. (2012).

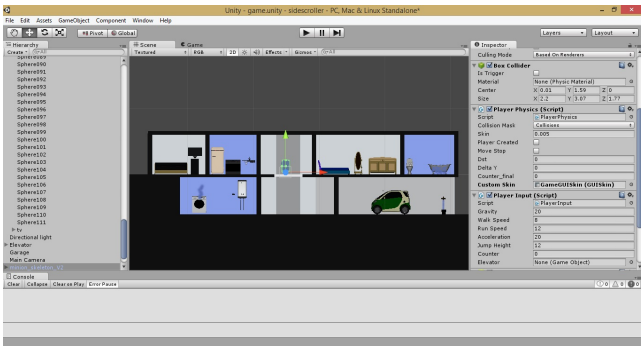


Fig. 9. Nível

Fig. 10. Sistema de partículas

http://blog.hubspot.com/blog/tabid/6307/bid/33423/19-Reasons-You-Should-Include-Visual-Content-in-Your-Marketing-Data.aspx.

Archive3D. 2012. Archive3D. (2012). http://archive3d.net/.

Blender Features. 2014. Blender. (2014). www.blender.org/features.

Blender for noobs. 2013. How to make a minion. (2013). https://www.youtube.com/watch?v=-2uY7rjhhMs.

Blender Guru. 2014. Blender Guru. (2014). www.blenderguru.com.

Construção Sustentável. 2013. Boas práticas para uma eficiente utilização da água. (2013).

David Williamson Shaffer et. al. 2004. Video games and the future of learning. (2004). http://www.academiccolab.org/resources/gappspaper1.pdf.

Jeremy Dean. 2013. *Making Habits, Breaking Habits: Why We Do Things, Why We Don't, and How to Make Any Change Stick*. a Capo Pr.

Dr. BJ Fogg, Stanford. 2011. Fogg Model. (2011). http://www.behaviormodel.org.

Charles Duhigg. 2014. *The Power of Habit*. Random House Trade Paperbacks.

FAO. 2013. Food wastage footprint. (2013). http://www.fao.org/docrep/018/i3347e/i3347e.pdf.

GameMaker. 2013. YoYo Games. (2013). https://www.yoyogames.com/studio.

GameMaker Community. 2013. YoYo Games. (2013). http://gmc.yoyogames.com/.

Habbit RPG. 2012. Habbit RPG. (2012). http://habitrpg.wikia.com/wiki/FAQ.

John Rose. 2008. Fewer Mechanics, Better Game. (2008).

John SeelyBrown et. al. 1989. Situated Cognition and the Culture of Learning. (1989).

Joshua Porter. 2013. Principles of user interface design. (2013). http://bokardo.com/principles-of-user-interface-design/.

Lauren Hall-Stigerts. 2013. How to Change Any Bad Habit Using Game Mechanics. (2013). http://www.bigfishgames.com/blog/how-to-change-any-bad-habit-using-game-mechanics/.

Pierre Coffin, Chris Renaud. 2010. Despicable Me. (2010). http://www.imdb.com/title/tt1323594/.

Rockstar. 1998. Grand Theft Auto. (1998). http://www.rockstargames.com/grandtheftauto.

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

Stanford Encyclopedia of Philosophy. 2000. Behaviorism. (May 2000). http://plato.stanford.edu/entries/behaviorism/2.

Super Mario, Shigeru Miyamoto. 1985. Super Mario. Nintendo. (1985). http://mario.nintendo.com/.

Unity Community. -. Unity. (-). http://gmc.yoyogames.com/index.php?act=idx.

Unity documentation. 2012a. Mesh Collider. (2012). http://docs.unity3d.com/Documentation/Components/class-MeshCollider.html.

Unity documentation. 2012b. Unity Script guide. (2012). http://docs.unity3d.com/Documentation/ScriptReference/Physics.Raycast.html.

Unity documentation. 2013. Box Collide. (2013). http://docs.unity3d.com/Documentation/Components/class-BoxCollider.html.

Unity Technologies. -. Unity. (-).

UnityGUI. 2012. GUI Scripting guide. (2012). https://docs.unity3d.com/Documentation/Components/GUIScriptingGuide.html.