Universidade de Brasília
Instituto de Exatas
Departamento de Ciência da Computação

Jogos Ubíquos Reconfiguráveis

Da Concepção à Construção

Fabricio Nogueira Buzeto

Brasília
2015

Universidade de Brasília
Instituto de Exatas
Departamento de Ciência da Computação

Jogos Ubíquos Reconfiguráveis

Da Concepção à Construção

Fabricio Nogueira Buzeto

Brasília
2015

Universidade de Brasília
Instituto de Exatas
Departamento de Ciência da Computação

Jogos Ubíquos Reconfiguráveis

Da Concepção à Construção

Fabricio Nogueira Buzeto

Tese apresentada como requisito final
para conclusão do Doutorado em Computaçãono PPGINF

Orientador
Prof. Dr. Ricardo Pezzuol Jacobi

Brasília
2015

Universidade de Brasília
Instituto de Exatas
Departamento de Ciência da Computação

Jogos Ubíquos Reconfiguráveis

Da Concepção à Construção

Fabricio Nogueira Buzeto

Tese apresentada como requisito final
para conclusão do Doutorado em Computação no PPGINF

Prof. Dr. Ricardo Pezzuol Jacobi (Orientador)
CIC/UnB

Prof. Dr. André Luiz Battaiola
Departamento de Design/UFPR

Prof. Dr. Esteban Walter Gonzalez Clua
INE/UFSC

Prof.ª Dr.ª Genaina Nunes Rodrigues
CIC/UnB

Prof. Dr. Rodrigo Bonifacio de Almeida
Instituto de Computação/UFF

Prof.ª Dr.ª Célia Ghedini Ralha
Coordenadora do Programa de Pós-graduação em Doutorado em Computação

Brasília, 15 de Dezembro de 2015

Dedicatória

Dedico este trabalho a todos que acreditam na força transformadora que a tecnologia possui sobre a vida das pessoas. Em especial àqueles que dedicam suas vidas para que tal impulso seja dado na direção do avanço da humanidade.

Agradecimentos

A tarefa de expressar a minha gratidão de forma completa a quem devo é tão difícil quanto provar que P=NP. Apesar deste trabalho possuir apenas um autor em sua definição, ele agrega a contribuição de uma vasta gama de contribuintes que vão muito além das citações e este humilde agradecimento. Soma-se ainda a este desafio o fato deste trabalho coroar o fim de uma trajetória iniciada ainda em 2002, com o início da minha vida acadêmica. É com todas estas dificuldades em mente que me arrisco na tarefa de agradecer, sem a pretensão de fazê-lo de forma completa.

Primeiramente coloco a minha gratidão a Deus. Verdadeiramente ubíquo e esteve sempre ao meu lado me guiando na montanha russa que tem sido o caminho até aqui.

Minha gratidão a Pah, quase co-autora responsável pela difícil tarefa de revisar meus textos. Sempre compreensiva e dedicada, me alimentando e suportando por todo esse tempo. Tamanha foi sua paciência que no inicio da jornada era apenas uma amiga e hoje é minha esposa e companheira pelo resto da vida. Agradeço ainda aos meus pais e irmãos que me deram suporte, sofreram juntos e me apoiaram, apesar das minhas recusas de explicar o que eu fazia afinal.

Agradeço à todos que fizeram parte do UnBiquitous, grupo que fundei e foi base deste trabalho. Primeiro aos meus orientadores Carla e Ricardo que em 2007 apostaram em um aluno desconhecido e por fim me carregaram pelos degraus da carreira acadêmica. A eles não só agradecimento, como também respeito e amizade. Não posso esquecer do Alê Gomes, pivô que me colocou nessa estrada ubíqua. Não fosse seu entusiasmo, minhas trilhas acadêmica e profissionais teriam sido bem diferentes. Além destes, tantos outros que contribuiram com a pesquisa e que me vem em memória : Estevão Passarinho, Marcelo Bassani, Ana Ozaki, Luciano Santos, Matheus Pimenta além de muitos outros. Destaque especial ao professor Tiago Barros, rebelde acadêmico e contribuinte por trazer uma visão prática e bela a minha pesquisa.

Não tem como esquecer meus sócios da Intacto e Qual Canal, que por mais que tenham sofrido com a minha escolha, no fim a suportaram. Em especial o Carlos Botelho, companheiro de Trabalho de Graduação e motivador para o meu início no programa.

Agradeço ainda a UnB e o Departamento de Ciência da Computação. Após mais de uma década é impossível não ter um carinho especial pela instituição apesar de todos os seus problemas e discordâncias. Com destaque aos professores que marcaram meu caminho como Maria Emília, João Gondim e Alba Melo.

Por fim agradeço a CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) e CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) pelo suporte financeiro fornecido ao longo desta pesquisa.

A quem quer que eu tenha esquecido que não fique ressentido, saiba que o espaço aqui é pouco mas a gratidão é ilimitada.

Conteúdo

Lista de Figuras

Lista de Tabelas

Abstract

The development of ubiquitous computing arises from the increasing number, diversity and pervasiveness of computational devices in our everyday lives. In addition to new interfaces and mobility of devices, these factors allow the development of systems more aligned with the ideal of invisible computing embedded in the user routine. It is by exploring *ubicomp* principles that electronic entertainment, specifically games, brought to life the concept of ubiquitous games, also known as pervasive games or just *ubigames*. These are focused on creating imaginary environments closely embedded to the real world aiming for greater immersion and engagement from players.

Although such games bring new characteristics of entertainment to the table they still lack of innovation regarding new game designs. This is clear by the fact that the usage of contextual information remains limited to just a few, restricting the possibilities of interaction. In face of the innovation potential of new game designs possible through ubigames, this work first dives in the study of alternatives that take advantage of these capabilities. This way more variability of design and exploration of game aspects can be introduced in the ubiquitous games scenario. This ends in the first contribution of this work, a definition of a subset of ubigame called "Reconfigurable ubigame", which brings reconfigurability (the ability to change and adapt in real time according to the context) as central part of the game design. This makes possibilities more explicit to designers when creating new games, pointing dimensions and characteristics relevant to explore in a creative way solutions for a more immersive and engaging entertainment.

Developers are another group involved in this scenario, being involved in the challenges of bringing these concepts to life. The literature lists a few platform focused on easing the task of developing such games. Although, none of them satisfy in a broader and complete way the technical requirements involved in such task. Among these requirements can be highlighted the heterogeneity of technologies, mobility, context sensitivity, spontaneous integration and additional tooling for games themselves. The second contribution of this work is the *uOS* game development platform that satisfies these requirements. Additionally, it provides means of using code mobility in the environment, allowing the dynamic reconfiguration of game behaviour in runtime.

Both contributions were evaluated through the creation and development of games that makes use of the defined concept of reconfigurable ubigames. The different aspects incorporated in these games, their description and analysis is presented as result of this thesis, as part of the main goals of this work.

Keywords: Ubiquitous Computing, ubicomp, Code Mobility, Pervasive Games, Ubiquitous Games, ubigames, Reconfigurable Games

Resumo

O aumento da diversidade e pervasividade dos dispositivos computacionais no cotidiano das pessoas é uma condição básica para o desenvolvimento da Computação Ubíqua - *ubicomp*. São fatores que, associados à novas interfaces e à mobilidade dos dispositivos, permitem o desenvolvimento de sistemas progressivamente mais próximos do ideal de computação invisível, incorporada ao dia a dia do usuário. A exploração dos princípios da ubicomp no ramo dos jogos eletrônicos deu origem aos jogos ubíquos ou pervasivos - também conhecidos por *ubigames*, que se interessam pela criação de universos lúdicos, intrinsecamente conectados ao mundo real, na busca de um maior engajamento e imersão dos jogadores.

Embora tragam características da *ubicomp* ao contexto do entretenimento eletrônico, observa-se nestes jogos relativamente pouca inovação quanto a exploração de novos *game designs*, com o uso restrito das informações de contexto e uma limitada exploração das possibilidades de interação. Considerando-se o potencial de inovação possibilitado pelos *ubigames* e as possibilidade de novos *game design*, a primeira contribuição deste trabalho é um estudo das alternativas de interação de forma a introduzir maior variabilidade e mobilidade de dispositivos no ambiente e seu reflexo nos diferentes aspectos dos jogos. Este estudo culmina com a proposta de uma definição de um tipo particular de *ubigame*, a saber, os jogos ubíquos reconfiguráveis. Onde reconfigurabilidade diz respeito a adaptação do jogo a configuração do ambiente em tempo real. Definição esta que torna explícita algumas possibilidades de inovação auxiliando *game designers* na concepção de novos jogos ubíquos, apontando dimensões relevantes para a exploração criativa do espaço de soluções na direção de um entretenimento mais envolvente e imersivo.

Neste espaço, soma-se ainda o desafio técnico de quem materializa estes jogos, os desenvolvedores. Diversas plataformas com o intuito de auxiliar nesta tarefa podem ser encontradas na literatura. Porém, nenhuma delas atende de forma abrangente os requisitos tecnológicos necessários a estes jogos, em particular os aspectos relacionados à heterogeneidade das tecnologias, mobilidade, sensibilidade ao contexto e integração espontânea de dispositivos. Neste trabalho foi desenvolvida uma plataforma para apoio à construção de jogos ubíquos em geral, a *uOS*, contemplando de forma transversal tais aspectos. Adicionalmente esta plataforma fornece ainda recursos de mobilidade de código, permitindo o ajuste dinâmico do comportamento dos jogos através da transferência de informações de execução.

Ambas contribuições foram avaliadas através da concepção e desenvolvimento de jogos que exploram conceitos definidos pelos jogos ubíquos reconfiguráveis e que utilizam a *uOS*. Os jogos obtidos incorporam diferentes aspectos da *ubicomp*, e sua descrição e análise é apresentada nos resultados da tese, demonstrando que os principais objetivos propostos neste trabalho foram atingidos.

Palavras-chave: Computação Ubíqua, ubicomp, Mobilidade de Código, Jogos Pervasivos, Jogos Ubíquos, ubigames, Jogos Reconfiguráveis

Capítulo 1

Introdução

A computação ubíqua [93] tem sua origem na visão de uma sociedade onde os dispositivos computacionais cada vez mais estão presentes no cotidiano das pessoas. Ao longo das últimas décadas, tem-se vivido de forma clara a transição entre a era da computação pessoal, na qual cada pessoa possui um dispositivo exclusivo e de uso pessoal, para a computação pervasiva, onde estes se espalham ao nosso redor [94]. Isso é ainda mais evidente quando se observa a imensa variabilidade de dispositivos que têm sido incrementados de poder computacional. Exemplos incluem tanto aqueles de uso compartilhado (TVs, geladeiras e carros) como os de uso individual (relógios, telefones e óculos). Apesar disso, a visão almejada pela *ubicomp*, onde toda essa computação dispersa no ambiente é utilizada de modo transparente aos olhos do usuário[1], ainda não se tornou realidade [18].

Mesmo considerando que o estado atual da computação não se encontra alinhado por completo com o aquele objetivado pela *ubicomp*, grandes avanços têm sido alcançados, tanto em aplicações quanto especialmente em tecnologia. Esses avanços serviram de motivação para o emprego dos conceitos e tecnologias consideradas ubíquas no ramo dos jogos eletrônicos, dando origem aos *jogos ubíquos*, ou apenas *ubigames*. O propósito destes, é utilizar a tecnologia na criação de uma conexão mais intensa entre os elementos do mundo real e o universo virtual a ser vivido [15]. Característica esta que busca aumentar tanto a imersão quanto o engajamento do usuário, expandindo a visão da narrativa experimentada para além do dispositivo utilizado.

Diversos títulos têm sido desenvolvidos explorando a liberdade que esse tipo de jogo permite. A diversidade é tão grande que hoje o desenvolvimento de *ubigames* não se restringe apenas à academia, incluindo também a indústria de entretenimento eletrônico. Este crescimento é fortemente impulsionado pela propagação de dispositivos móveis (como *tablets* e *smart phones*) e novos sensores atrelados aos mais populares consoles. Dentre os estilos que mais se beneficiaram dessas tecnologias estão os jogos baseados em localização (*LBGs*) e de realidade alternada (*ARGs*), que representam grande parte dos títulos disponíveis. Apesar de positiva, esta predominância é restritiva ao *game design* dos títulos, limitando-os ao uso de poucos elementos possíveis [83]. Soma-se ainda que a parte dominante dos títulos desenvolvidos pela academia possuem seu foco voltado na aplicação

[1] A este tipo de cenário é dado o nome de *ambiente inteligente* (do inglês *smart space*), devido a sua capacidade de agir em prol do usuário observando o contexto que o cerca.

tecnológica ou, quando focados na experiência do usuário, restringem-se "jogos eventos"[2] [63]. Isso ocorre em detrimento do surgimento de "jogos serviço"[3], que são o foco dessa indústria.

Do ponto de vista técnico, o desenvolvimento de aplicações ubíquas sempre envolveu uma série de desafios [31]. Para jogos ubíquos se destacam as questões de heterogeneidade, mobilidade, integração espontânea e sensibilidade a contexto[4]. Diversas iniciativas foram desenvolvidas com o propósito de superar os desafios no contexto de *ubigames* e, apesar de serem encontradas plataformas que favoreçam tanto jogos gerais quanto gêneros específicos, estes quatro desafios continuam não atendidos em sua completude.

Neste cenário vemos dois públicos de profissionais formam o foco desta pesquisa: *game designers* e *game developers* (projetistas e desenvolvedores de jogos). Com o foco no projeto e implementação de jogos ubíquos, esta pesquisa tem por motivador avaliar qual o ferramental disponível, quais as lacunas e como estas podem ser atendidas do ponto de vista computacional. Em especial, como estas lacunas influenciam na proposição de *game designs* distintos dos existentes e que façam melhor uso das capacidades e possibilidades do ambiente de forma transparente. Levando a pergunta de pesquisa deste trabalho:

> Quais as lacunas ferramentais que desenvolvedores e projetistas de jogos enfrentam no contexto dos jogos ubíquos e como estas podem ser atendidas com foco na transparência do ambiente?

Com relação aos projetistas, observa-se que limitações quanto a variabilidade e inovação no *game design* dos jogos ubíquos expressa-se tanto na dimensão de suas mecânicas (dominada por caças ao tesouro) bem como no uso de poucas informações de contexto ou forma de interação. A predominânica dos *ARGs* e *LBGs* é um dos sintomas do foco quase exclusivo do uso de poucos dados do ambiente, usualmente se restringindo a localização do usuário. Mais ainda, nenhum título encontrado nesta pesquisa faz uso da adaptabilidade e inteligência característica da *ubicomp* nas mecânicas de jogos. Isto fica claro onde a maior parte dos jogos consiste na simples adpatação de títulos já existentes para um cenário mais tátil. Mesmos os títulos que se superam em prover experiências diferentes e inovadoras acabam por se limitar a jogos eventos que possuem nenhuma longevidade entre os jogadores. Promover emergência interativa [38] entre estes títulos é uma necessidade pois possibilitaria o surgimento de novas experiências a partir das interações dos próprios jogadores. Permitindo assim que os usuários influenciem o resultado e o andamento dos jogos, prolongando sua existência. Desta forma os objetivos específicos relacionados ao projeto de jogos ubíquos são:

- Levantar as ferramentas e metodologias aplicadas na elaboração do *game design* de jogos ubíquos.
- Levantar as deficiências envolvidas nos projetos de jogos ubíquos.

[2]Estes são jogos de duração determinada, existindo apenas enquanto seus eventos são executados e deixando de existir em seguida.

[3]Do inglês "*service games*", são jogos que possuem grande longevidade, permitindo que seus jogadores tenham experiência do jogo quando desejarem.

[4]Isto não implica que demais desafios não se apliquem ao contexto dos *ubigames*, apenas que estes quatro representam a base necessária para a operacionalização de um título deste tipo.

- Definir uma metodologia ou ferramental que auxilie projetistas de jogo a superar as deficiências encontradas.

Deve-se considerar ainda, que uma plataforma com o intuito de viabilizar o desenvolvimento de *ubigames* deve fornecer ferramentas que possibilitem tanto a integração de dispositivos (desafios da *ubicomp*) como a confecção de jogos. Com relação a integração de dispositivos pode-se ressaltar quatro desafios relacionados a aplicações ubíquas e um relacionados a jogos. sendo eles: Sensibilidade a Contexto, Heterogeneidade, Integração Espontânea, Mobilidade e Componentes de Jogos. Entende-se por sensibilidade a contexto a capacidade de observar e compreender os recursos do ambiente e as interfaces de acesso aos mesmos. Heterogeneidade diz respeito a permitir a integração de diversos dispositivos que utilizem plataformas distintas de hardware e software além de redes de comunição diferentes. Permitir que novos tipos de dispositivos e recursos sejam acessíveis, em tempo real e sem a necessidade de intervenção, faz parte do que se entende por integração espontânea. Dispositivos podem entrar e sair do ambiente de forma inesperada, como parte do se espera do desafio de mobilidade. Por fim, fornecer componentes de jogo que controlem o ciclo de vida do jogo e forneça componentes reusáveis, simplificando seu desenvolvimento. Neste cenário, os objetivos específicos relacionados ao desenvolvimento de jogos ubíquos são:

- Levantar as ferramentas e metodologias aplicadas na implementação de jogos ubíquos.
- Levantar como estas ferramentas buscam atender aos desafios envolvidos no desenvolvimento deste tipo de jogo.
- Construir uma ferramenta que atenda a estes desafios.

Como resultado final desta tese foram elaboradas duas soluções: os Jogos Ubíquos Reconfiguráveis e a Plataforma de Desenvolvimento de Jogos Ubíquos *uOS*. O primeiro como um subconjunto dos *ubigames*, como forma de guiar os projetista na criação de *game designs* que se auto-adaptam às mudanças de configuração do ambiente em tempo real. Alavancando assim uma das principais características da *ubicomp*, a integração de dispositivos e informações do ambiente, aumentando as possibilidades de interação. O segundo resultado, a plataforma *uOS*, fornece um conjunto de soluções que atende aos desafios destacados da *ubicomp* ao mesmo tempo que fornece ferramentas especializadas para o desenvolvimento de jogos ubíquos e em especial reconfiguráveis.

Este documento se encontra organizado da seguinte maneira. No capítulo 2 é apresentada em detalhes a área de jogos ubíquos, seus conceitos, desafios, estado atual e plataformas de desenvolvimento existentes. O conceito de jogos ubíquos reconfiguráveis é introduzido no Capítulo 3, onde pode-se encontrar sua definição e gradação, bem como exemplos de jogos existentes. O Capítulo 4 explica de forma detalhada o conjunto de soluções que compõem a plataforma *uOS* para o desenvolvimento de jogos ubíquos. A aplicação tanto dos conceitos de jogos reconfiguráveis como da plataforma construída é reportada no Capítulo 5 através de onze títulos desenvolvidos. Por fim, o Capítulo 6 apresenta as considerações finais e intenções futuras decorrentes. De forma complementar, o Apêndice A fornece um compêndio de jogos ubíquos utilizados no estudo motivador deste projeto.

Capítulo 2

Jogos Ubíquos

> *"Nenhum outro setor experimentou um crescimento tão explosivo como a indústria de jogos para consoles e computadores. Nossos produtores criativos e sua força de trabalho talentosa continua a acelerar o avanço e pioneirismo em novos produtos, rompendo os limites existentes para despertar experiências de entretenimento. Estas inovações impulsionam uma conectividade incrementada entre os jogadores, alimentam a demanda por produtos, e encorajam o progresso e a expansão de uma diversificada base consumidora."*
>
> – Michael D. Gallagher, presidente e CEO da *Entertainment Software Association*

É desta forma que se inicia o relatório de 2013 [8] acerca da indústria de jogos eletrônicos nos Estados Unidos. Uma indústria movida por inovação que, desde seu surgimento na década de 1970, apresenta um crescimento impressionante. Fato ainda mais evidente quando se constata que atualmente o mercado destes jogos supera em três vezes o da indústria do cinema [1].

Movida por uma busca constante por novidades, a indústria de jogos eletrônicos tem empurrado a tecnologia além de seus limites procurando trazer novas experiências aos seus usuários. No ramo de computadores pessoais e consoles, placas de vídeo e som cada vez mais poderosas e fidedignas têm sido desenvolvidas a fim de atender jogadores e produtores interessados em uma realidade virtual cada vez mais representativa do mundo real. Porém a inovação não se limita apenas aos tradicionais *PCs* e consoles[1], incluindo os dispositivos móveis, celulares e outras formas de se jogar. Desde 2013 metade da renda relativa a venda de jogos eletrônicos é consumida através de plataformas distintas dos consoles e dos *PCs* [8]. É nesta busca por novas formas de prover entretenimento através de jogos que a computação ubíqua oferece uma nova porta de inovação, os jogos ubíquos.

O propósito da computação ubíqua é desonerar a atenção do usuário tirando seu foco dos recursos computacionais a sua volta. Sua busca por estabelecer um mundo onde a relação com a tecnologia seja "calma" não é naturalmente alinhada com os objetivos dos jogos e a diversão que eles proporcionam. Diferentemente da relação utilitária que a Computação Ubíqua nos apresenta, a relação que se tem com jogos explora aspectos emocionais. Por esta razão que Weiser and Brown [95] afirmam que jogos não são aplicações adequadas à ubicomp já que "um jogo calmo é de pouco uso sendo que seu propósito é excitar".

[1] Onde podemos também considerar os *arcades*.

Porém, a busca por uma computação "calma", como almejada pela *ubicomp* levou à criação de diversas tecnologias e conceitos que permitem mesclar de maneira mais abrangente componentes computacionais a alguns elementos do dia a dia. Essa mistura despertou o interesse em utilizá-los a fim de expandir o universo de interações e metáforas existentes nos jogos, motivando assim que os primeiros experimentos na área ocorressem. Em um dos primeiros trabalhos que abordam este tema, Björk et al. [15] define jogos ubíquos como aqueles que permitem que tanto jogadores online como jogadores no mundo real compartilhem uma mesma experiência. Em tais jogos, o contexto presente no mundo real deve ser influenciado pelo "jogar" através de dispositivos computacionais.

Assim como a computação ubíqua que possui múltiplas definições que se sobrepõem, como Computação Móvel, Computação Pervasiva e Internet das Coisas [2, 91], os jogos ubíquos também possuem uma variedade de definições e nomes. A seguir são apresentados alguns dos principais termos empregados na academia. É possível observar a ausência de consenso, existindo sobreposição entre algumas delas.

- **Jogos Ubíquos** [15]: *"[Jogos Ubíquos] permitem que tanto jogadores online como jogadores no mundo real compartilhem uma mesma experiência"*. Sendo uma das primeiras definições publicadas na literatura, estas mostram o intuito dos autores em explorar a interação entre jogadores do mundo real com aqueles no virtual. Nota-se a relação de exclusividade, onde jogadores online apenas vivenciam o mundo real através de uma representação virtual enquanto aqueles no mundo real tem acesso a dispositivos que realizam esta conexão.

- **Jogos de trans-realidade** [46]: *"Jogos trans-realidade permitem ao jogador viver mundos virtuais paralelos ao mundo real"*. Diferencia-se da definição anterior pois aqui não existe mais a restrição onde jogadores online devem ser incluídos na experiência. Estes jogos tem seu foco em permitir a vivência de "mundos virtuais paralelos" ao mundo real. Seus jogadores devem possuir a habilidade de alternar entre ambas as realidades usando os recursos computacionais disponíveis.

- **Jogos Sensíveis a Contexto** [56]: *"Jogos Sensíveis ao contexto reagem as mudanças que ocorrem no ambiente a sua volta"*. Possui uma definição ampla abrangendo todos os jogos que possuem a habilidade de se adaptar a alterações no ambiente. Tais jogos reagem a mudanças climáticas, na localização do usuário ou mesmo informações biométricas (por exemplo, o esforço físico).

- **Jogos Pervasivos** [87]: *"Jogos Pervasivos utilizam a localização do jogador como entrada"*. Define-se como jogos que permitem aos seus usuários mobilidade e que se utilizam da sua geolocalização como informação de contexto. Também conhecidos como *LBGs* (*Location Based Games*).

- **Jogos de Realidade Misturada** [16] : *"São aqueles que utilizam a tecnologia para criar experiências que transformam as relações entre pessoas e lugares, misturando o real e virtual"*. Se incluem junto aos gêneros de jogos de realidade aumentada e alternada, onde o mundo real é usado como cenário para uma experiência virtual.

A diversidade de definições expõe o não alinhamento das intenções dos diversos autores com relação ao seu entendimento desta categoria de jogos. Em alguns casos temos definições abrangentes, em outros, mais específicas e restritas. Apesar disto, estes convergem

em buscar modos de utilizar a tecnologia que cada vez mais se espalha pelo dia a dia do usuário de forma a torná-la parte da experiência de jogo. Em particular uma definição que consideramos mais alinhada a idéia de *ubigames* é a proposta por Guo et al. [44]:

> Consiste em um novo ramo do entretenimento, sendo então um de seus produtos. Combinando elementos reais e virtuais, permitindo que usuários interajam com seus arredores durante o jogo. Deste modo pessoas podem se envolver de forma completa com os **Jogos Pervasivos** obtendo uma experiência de jogo aprimorada.

Portanto, *ubigames* se utilizam da combinação de elementos do mundo real e virtual de forma a aprimorar a experiência e aumentar o engajamento dos jogadores. Esta é uma das características que fazem com eles se diferenciem de outras "aplicações ubíquas". Enquanto estas buscam usar os recursos computacionais do ambiente de forma a poupar a atenção do usuário permitindo que se concentrem em suas atividades, aqueles destacam elementos do mundo real (sob o viés lúdico) aos olhos do jogador. Ambos tem por objetivo permitir que o usuário tenha uma maior imersão na atividade sendo realizada. Porém, em uma aplicação ubíqua isto acontece através de uma maior transparência e invisibilidade dos elementos computacionais. Já em um jogo ubíquo, estes elementos são realçados através das metáforas do jogo, em busca de um maior engajamento com o universo lúdico proposto.

Exemplos de como jogos ubíquos realizam esta mescla de elementos podem ser encontrados na Seção 2.1. A Seção 2.2 apresenta propostas que buscam classificá-los e categorizá-los. Estas definições como propósito auxiliar o entendimento dos elementos que compõem, facilitando sua concepção e criação deste tipo de interação. Uma discussão acerca do estado atual dos jogos ubíquos é retratada na Seção 2.3 enquanto os desafios enfrentados nesta área são expostos na seção 2.4. Por fim, as Seções 2.5 e 2.6 apresentam algumas plataformas de desenvolvimento de jogos ubíquos e uma comparação de suas características.

2.1 Exemplos de Jogos Ubíquos

Pesquisas em jogos ubíquos já vêm sendo realizadas há mais de uma década, tendo como fruto uma vasta gama de gêneros e títulos desenvolvidos. Em meio a estes se encontram tanto esforços vindos da academia como da indústria. A seguir são apresentados três títulos representativos deste tipo de jogo. É possível observar que são jogos que carregam consigo elementos comuns a outros títulos de entretenimento eletrônico porém, no caso deles elementos do mundo real são embarcados em seu funcionamento. Estes elementos correspondem às capacidades disponibilizadas pela computação ubíqua à tais jogos. Complementar a estes títulos, pode se encontrar uma lista extensiva de jogos ubíquos e tecnologias a eles relacionadas no Apêndice A.

O jogo "*Uncle Roy All Around you*" [13] (Figura 2.1) consiste em um "jogo evento". Suas sessões tiveram como cenário uma das praças da cidade de Londres, na Inglaterra, em 2003. Voluntários eram convidados a participar da caçada ao seu perdido "tio *Roy*" e para isso lhes era dado um *PDA* contendo o mapa da região. Para alcançar seu objetivo, os jogadores deviam seguir uma série de pistas que os levavam a locais espalhados pelas imediações. Nesta tarefa, jogadores contavam com a ajuda de outros participantes online

(a) Jogador interagindo na interface virtual do jogo.

(b) PDA utilizado como interface do jogo.

(c) Jogador nas ruas em busca de pistas.

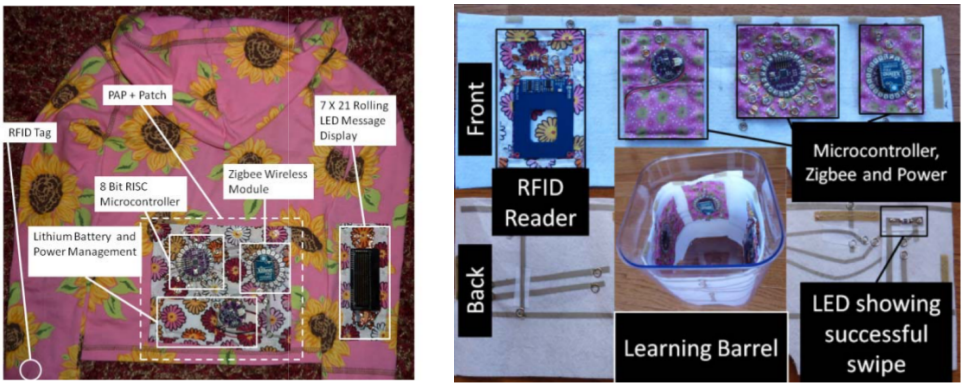
Figura 2.1: Jogo Uncle Roy All Around You [13].

que poderiam contribuir com mensagens de texto ou áudio. Neste teste ficou claro como uma mesma experiência, sob diferentes pontos de vista, modifica a vivência do jogo. Mais ainda, com o compartilhamento desses pontos de vista em tempo real, a vivência se torna ainda mais intensa. Além disso, também se verificou como os usuários reagiam a um sistema de localização onde eles mesmos informavam suas posições (não coletadas por nenhum tipo de sensor).

"*Hoodies and Barrels*" [7] (Figura 2.2) tem por objetivo ajudar crianças a aprenderem sobre matemática brincando com as noções de dimensão e distância em duas variações do jogo de pega-pega. Para isso são utilizados dois elementos de jogo: jaquetas e barris. As jaquetas são artefatos vestíveis utilizados pelas crianças. Nelas encontram-se etiquetas RFID[2] e um mostrador de *LED* que se comunicam através de *ZigBee*[3] com uma central. Já os barris possuem uma antena *RFID* que permite estimar a distância das crianças repassar esta informação à central. Em uma primeira modalidade do jogo, o professor estabelece desafios a serem resolvidos pelas crianças. Estes desafios envolvem problemas lógicos que são registrados antes da sessão de jogo. Um exemplo de desafio seria que todas as crianças devem se esconder atrás de um objeto com mais de 50 metros cúbicos. No início da partida, o jogador recebe a restrição e deve então encontrar seus amigos. Em uma segunda modalidade, as crianças se escondem e definem dicas de sua localização. O jogador "caçador" deve então seguir as dicas, ativadas por proximidade, para encontrar seus amigos.

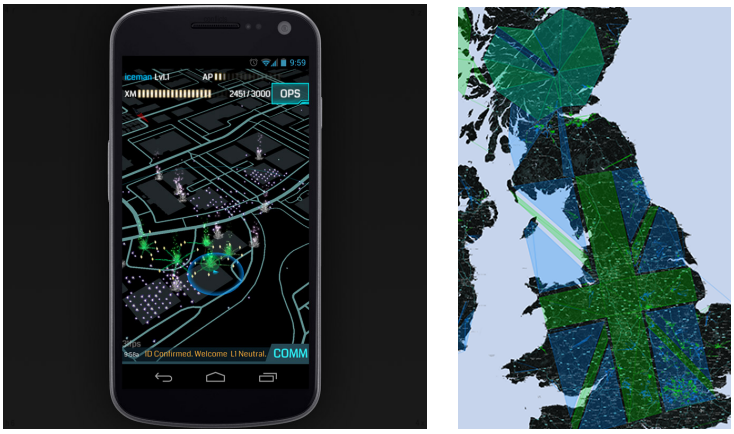
[2]Etiquetas de identificação por rádio freqüência, do inglês *Radio Frequency ID*

[3]Tecnologia de comunicação de rádio de baixo consumo energético. `http://www.zigbee.org/`



(a) Componentes do casaco. (b) Componentes dos barris.

Figura 2.2: Elementos do jogo Hoodies and Barrels [7].



(a) Interface de jogo. (b) Criação feita por jogadores na Inglaterra.

Figura 2.3: Exemplos de interação do jogo Ingress.

O gênero conhecido como *ARGs* (Jogos de Realidade Alternada, do inglês "*Alternate Reality Games*") propõem a criação de mundos virtuais que usam o mundo real como seu tabuleiro. O jogo Ingress[4] (Figura 2.3) se enquadra nesta categoria. Em sua história, uma estranha energia está sendo detectada em diversos pontos do globo, e esta é originada de portais espalhados pelo mundo. Os jogadores devem escolher entre duas facções: a resistência, que acredita que tal energia é maligna e deve ser restringida; e os iluminados, que acreditam que esta energia deve ser estudada e explorada. Os jogadores então devem criar e capturar portais pelo mundo, estabelecendo suas estratégias em cada local. De forma colaborativa, o jogo também permite que seus participantes participem de fóruns e encontros onde a história do mesmo é cocriada pelos membros da comunidade do jogo. Tendo sido lançado ao final de 2012, e ainda em andamento, estima-se um número de jogadores próximo de 7 Milhões [5].

Estes exemplos ilustram como jogos ubíquos vêm sendo desenvolvidos na academia e

[4] `http:://www.ingress.com`
[5] `https://play.google.com/store/apps/details?id=com.nianticproject.ingress`

na indústria explorando as liberdades que este tipo de jogo oferece. Uma gama maior de títulos e tecnologias utilizadas é apresentada no Apêndice A.

2.2 Classificação de Jogos Ubíquos

Com uma vasta e crescente diversidade de títulos de jogos ubíquos constata-se que não só existe um interesse por este tipo de entretenimento, como também pelas possibilidades criativas proporcionadas por eles. Com a intenção de melhor entender este ramo do entretenimento, bem como auxiliar no desenvolvimento de novos jogos, alguns autores criaram classificações para os *ubigames*. Aqui são apresentadas três destas classificações que evidenciam a relação dos jogos (e suas interações) com a tecnologia. São elas: por propósito ou objetivo, de Mcgonigal [61]; por grau de flexibilidade, de Guo et al. [44]; e por características, de Valente et al. [89] [90].

2.2.1 Classificação por Objetivo

Esta classificação [61] divide os jogos ubíquos de acordo com três distintas metáforas acerca do ato de jogar: a *colonização*, a *ruptura* e a *ativação* através do jogar. Cada uma destas metáforas possui suas próprias características bem como estratégias acerca de seus objetivos. Porém, todas buscam de alguma maneira redefinir a relação entre os jogos e a vida cotidiana. Desta forma, apesar de distintas, estas metáforas podem coexistir em um mesmo jogo. De maneira simplificada, estas podem ser definidas como:

- A **colonização** através dos jogos se dá pela crença que o instinto lúdico pode conquistar qualquer objeto tecnológico. Este lado lúdico pode ser usado como impulso facilitador na adoção de novas tecnologias, trazendo maior aceitação das mesmas.
- A **ruptura** dos jogos se dá quando estes se misturam ao cotidiano das pessoas, influenciando sua realidade e suas relações sociais. Mesmo assim, deve-se permitir que o usuário escolha quando e a quem estão visíveis suas ações.
- Características normalmente ignoradas em objetos do cotidiano podem ser **ativadas** de forma a criar um situação lúdica. Seu propósito é mudar a forma como estes objetos são vistos, permitindo que o jogo ocorra ao modificar a interpretação do contexto onde está inserido.

Jogos criados com o propósito de *colonizar* respondem a duas questões: se o propósito da computação ubíqua é permear todos os aspectos do cotidiano, jogos também serão afetados por essa mentalidade; se jogos tem a capacidade de usufruir das mais diversas tecnologias, as tecnologias ubíquas também devem ser utilizadas para o entretenimento. Em ambos os pontos de vista, a base está na tendência dos objetos do cotidiano serem incrementados de poder computacional. Desta forma é fácil imaginar que o "instinto lúdico" deve tirar proveito destas capacidades em busca de melhores interações e experiências. Parte dos jogos que se encaixam neste propósito são em si casos de uso para a análise das reações dos usuários a novas tecnologias. Isso ocorre na medida em que o ambiente lúdico auxilia a vencer a barreira ao seu uso inicial. Isso fica claro no caso do título "*Uncle Roy All Around You*" [13], onde a utilização de informações de localização

auto-informadas se mostrou suficientemente acurada e barata. Técnica similar aplicada hoje em aplicações gameficadas como *Foursquare/Swarm* [6].

Assim como a ubicomp busca tirar a computação do foco do usuário, jogos de *ruptura* buscam minimizar a distância entre o mundo real e o virtual. Desta maneira, tendem a ocorrer como pano de fundo ao mundo real, impondo ao jogador que realize tarefas com o que existe a sua volta. Os maiores exemplos deste tipo de propósito são os jogos de realidade alternada, como *Ingress* [7] e *GeoCaching* [8]. Eles se utilizam de diversas estratégias para permitir que os jogadores possam transitar de forma transparente entre o universo do jogo e a realidade que os cerca. Uma das estratégias utilizadas é permitir que jogadores interrompam suas atividades no jogo conforme desejam, selecionando quando é o melhor momento para retomá-lo. Outra característica é disponibilizar canais de comunicação[9] que permitem aos jogadores interagir entre si e cocriar o universo do qual o jogo faz parte. Uma forte característica destes jogos é que eles ocorrem de maneira simultânea com não-jogadores, que participam de forma invonluntária do jogo, sem o conhecimento do que está ocorrendo.

O propósito dos jogos que buscam *ativação* é questionar quais são as características que objetos e espaços necessitam para permitir a diversão. Sua inspiração vêm das brincadeiras de crianças que dão novo propósito a objetos e ambientes. Uma árvore vira uma base, dois chinelos um gol, um graveto uma varinha mágica. Tais jogos analisam o quê cada objeto pode proporcionar e os utilizam como metáforas para a criação de jogos. Isso permite que o jogador entenda seu funcionamento de forma mais intuitiva. Por exemplo, uma maçaneta deve ser rotacionada para que a porta se abra, caso em um jogo seja dada à maçaneta a função de abrir as comportas do navio, seu uso de dará de forma natural. Através do questionamento de quais características o ambiente possui, é permitido que estes jogos mudem o contexto que cerca os jogadores, impondo metáforas que criam o ambiente lúdico desejado. Dessa forma, jogos são apenas contextos que se utilizam de características acionáveis do ambiente. Estes jogos não possuem relação direta com o poder computacional, mas sim com o mundo a sua volta, ignorando assim a existência de um mundo virtual. Um exemplo é o jogo SuperBetter[10], que convida seus jogadores a se desafiarem em troca de congratulações de seus amigos. Apesar disto, estes jogos também podem se beneficiar da pervasividade da computação, como fica claro no uso do brinquedo *Color Flare* [10] que, servindo-se em apenas de um cilindro dotado de sensores e atuadores, viabiliza a criação de uma infinidade de brincadeiras.

2.2.2 Classificação Gradativa

Jogos eletrônicos tradicionais buscam aumentar o engajamento de seus usuários através da imersão destes em um mundo virtual próprio. Uma das formas com que isto é alcançado é criando uma distinção clara entre o mundo real e o mundo virtual, convidando o jogador a entrar no que é conhecido como "círculo mágico" [50]. Composto por três dimensões, este círculo impõe restrições ao usuário para que ele possa viver a realidade do jogo.

[6] `http://foursquare.com` e `https://www.swarmapp.com/`

[7] `http://ingress.com`

[8] `http://geocaching.com`

[9] Usualmente são utilizados *chats*, fóruns, *blogs* e redes sociais.

[10] `https://www.superbetter.com/`

Essa restrição se dá em três dimensões: Espaço, Tempo e Socialização. Jogos ubíquos propõem que estas dimensões sejam flexibilizadas, expandindo os limites impostos pelo "círculo mágico". Conforme as fronteiras do círculo se tornam mais nebulosas, uma quarta dimensão pode ser também analisada, a percepção [44]. Desta forma, jogos ubíquos podem ser vistos como uma evolução gradativa de como cada uma destas dimensões é flexibilizada. Para tanto, deve-se questionar acerca de um jogo, sobre : **onde** ocorre, **quando** ocorre, por **quem** é utilizado e **como** é experienciado.

O grau de **mobilidade** de um jogo, determina o quão flexível este é com relação a **onde** este pode ocorrer. A maioria dos consoles limita seus usuários a estarem na frente de suas TVs para tomar parte na experiência entregue. Espera-se que *ugigames* permitam ao seu público maior liberdade com relação a onde o título possa ser usufruído. Jogos móveis são um avanço nesta direção, porém conforme a própria mobilidade é inserida como elemento da mecânica, sua proximidade com o mundo real pode ser incrementada. Isso pode ser utilizado inclusive em casos que o contexto sejam ambientes fechados, onde a posição do jogador na sala pode ser um dado observado pela aplicação, indo até os *LBGs* onde a posição do usuário no ambiente externo é parte da mecânica.

A **temporalidade** determina **quando** é possível interagir com um jogo, onde quanto mais flexível, mais ubíquo o mesmo se posiciona. Sendo assim, títulos que possuem sessões bem limitadas são considerados menos ubíquos. O contrário se aplica a títulos que possibilitam ao usuário escolher quando participar do mundo proposto. Neste caso, o universo virtual criado continua a existir, porém cabe ao jogador escolher quando no seu cotidiano ele irá interagir.

Socialização através dos jogos não é exclusividade dos jogos ubíquos. Títulos multijogadores (*multi-player*) estão bem estabelecidos e permitem a escolha com **quem** se interage durante a sessão. Independente disso, jogos ubíquos podem buscar uma experiência social ainda mais ampla. Por esta razão, jogos com limitações na abrangência da experiência, por exemplo, se é apenas individual ou mesmo coletiva mas restrita (caso do uso de times), podem ser considerados menos ubíquos. O que se espera dos *ubigames* é que a experiência social no universo criado seja o mais abrangente possível, colocando os jogos massivos como um alvo a ser alcançado. Mais expressivamente, jogos que cultivam uma comunidade que participa da criação colaborativa deste mundo é considerada mais ubíqua.

Como um jogo interage com o ambiente é determinado por sua **percepção**. Isto inclui tanto a forma do sistema observar o ambiente, quanto a forma com que este estimula seus usuários. Sob tal viés, a coleta de informações usando formas bem estabelecidas como teclados, *mouses* e *joystics*, se mostra menos interessante. Espera-se de jogos ubíquos que se explore a utilização de outras informações e técnicas como comandos de voz, biometria e objetos interativos. Já no sentido da estimulação, recursos visuais (como telas) e auditivos (como caixas de som e fones) são bastante convencionais. Porém diversas técnicas buscam embutir no ambiente formas de prover informações. Dentre estas podemos citar projeções, realidade aumentada e superfícies táteis.

2.2.3 Classificação por Características

Jogos ubíquos podem tanto ser suportados pela tecnologia como focados na tecnologia. Os suportados pela tecnologia a utilizam como base para que a mecânica projetada possa

ocorrer. Já aqueles focados em tecnologia utilizam o efeito lúdico do jogar para avaliar como os usuários vão reagir à inovação proposta. Desta forma, a melhor forma de se avaliar um jogo ubíquo é através de um conjunto de características que permitam avaliar como o *software* deve reagir. Valente et al. [89] define quatro grupos de critérios que estabelecem as condições esperadas de jogos ubíquos. Destes, dois são obrigatórios: jogos devem utilizar dispositivos móveis e serem sensíveis ao contexto. Os dois restantes são complementares: jogos devem acessar dados remotamente e serem multijogadores.

A condição de se fazer uso de **tecnologias móveis** nos jogos se opõe a utilização de *PCs* e consoles tradicionais que confinam os usuários a um local estático. Já a **sensibilidade ao contexto** permite que tais jogos possam observar o ambiente a sua volta e reagir de acordo. Essa característica é necessária para que exista uma proximidade entre o universo do jogo e o mundo real.

A necessidade de interação entre **diversos jogadores** incrementa a experiência do jogo, permitindo que o círculo social do usuário seja envolvido e expandido. Interação esta que pode ocorrer de forma externa ao jogo, nos casos em que ambos se realizem no mesmo local. Por outro lado, o convívio pode acontecer dentro do próprio jogo, integrando jogadores de forma remota. Neste caso introduz-se a necessidade de uma **comunicação remota**.

Tendo em vista estes critérios, pode-se estabelecer um grupo de características a serem observadas no projeto de um jogo ubíquo. Elas são colocadas como desejáveis, desde que os critérios mínimos para que o jogo seja considerado um *ubigame*. Esta lista permite que ainda que o *game designer* tenha em mente quais pontos devem ser observados durante o projeto do jogo. São eles:

- Independência de dispositivo: Quanto maior a gama de dispositivos para os quais o jogo está disponível, maior o seu possível impacto.
- Política de tratamento de incerteza: As tecnologias utilizadas apresentam diversas incertezas e erros que influenciam na execução do jogo. Por exemplo, os dados de localização adquiridos por *GPS* podem ter taxas de erro consideráveis[11] ou mesmo não estarem disponíveis. Tratar estas incertezas envolve desde removê-las do sistema, corrigí-las ou mesmo explorá-las como parte da experiência proposta.
- Redefinição do ambiente local: O jogo deve estabelecer como os significados do mundo virtual serão impostos ao mundo real, seja através de uma configuração prévia, uso de objetos determinados, sensores ou mesmo com a participação dos usuários.
- Grau tangível dos objetos do jogo: O jogo pode utilizar seus componentes móveis de diversas maneiras. Por exemplo, um celular pode ser um simples terminal do jogo. Porém, explorando características de seus sensores (e outros objetos ao redor) pode criar uma interação mais próxima ao ambiente.
- Ritmo de jogo: A definição de como o jogo ocorre é influenciada diretamente pela escolha de tecnologias sendo utilizadas, e vice versa. Por isso, deve-se observar como cada interação será mapeada para definir se o ritmo será compatível com a mecânica almejada.

[11] Muitos aparelhos tem margens de precisão acima de 10 metros, dificultando o posicionamento de uma pessoa dentro de um grupo.

- Envolvimento de não-jogadores: Assim como objetos do cenário, pessoas que não fazem parte do jogo podem ser incluídas como parte do mesmo, caso seja interessante. Jogos como "*Uncle Roy All Around You*" [13] e *TMP* [36] exploram essa característica deixando seus jogadores mais engajados. Essa característica torna os limites entre o que faz parte do jogo ou não mais nebuloso, aumentando o grau de imersão.

- Usabilidade: Na concepção das metáforas e interfaces utilizadas, deve-se observar as características de usabilidade sendo aplicadas. Assim como outros softwares, deve-se esperar que o usuário tenha maior naturalidade e intuitividade ao interagir com o sistema construído.

- Interpolação com o cotidiano: Jogos que ocorrem como eventos artísticos requisitam que seus jogadores dediquem uma grande fatia de seu tempo em um único bloco. Já a maioria dos *ARGs*, permitem que o usuário estabeleça os melhores momentos para participar.

- Autonomia do Jogo: Alguns jogos necessitam de intervenção direta para que ocorram, seja preparando o ambiente antes da sessão de jogo ou adaptando-o durante sua execução. Já outros permitem que esta configuração ocorra de maneira automática ou mesmo que seja dispensável.

- Mobilidade: Apesar de mobilidade ser um critério necessário para que o jogo seja categorizado como ubíquo, o grau de sua exploração no jogo deve ser determinado. Por exemplo, jogos em ambientes fechados ou abertos, o uso de sensores de posição ou movimento e outras técnicas de rastreamento.

- Interação *cross-media*: Como demonstrado pelos *ARGs*, a interação com outras mídias pode aumentar as possibilidades de interação dos jogadores. Deve-se considerar como esta característica pode ser incorporada no jogo sendo desenhado.

- Persistência: Alguns jogos duram apenas pela tempo de sua sessão, outros carregam as informações de uma sessão para as subsequentes. Já jogos como *ARGs*, desconhecem o conceito de sessão, persistindo os dados ao longo de todas as interações que ocorrem.

- Interação Social: Permitir que os jogadores se comuniquem e interajam através de ferramentas dentro do próprio jogo como *chats*, ou externo ao mesmo, como fóruns e encontros presenciais. Estabelecer como essa comunicação vai afetar o jogo pode ir desde apenas uma coordenação entre grupos a interferência direta na narrativa.

- Conformidade com as definições sociais e físicas: Determinar quais os limites éticos do uso do ambiente e objetos, bem como preocupação com a privacidade deve ser avaliado.

- Conectividade: No caso de jogos que dependem de comunicação remota, a influência do meio usado deve ser considerado na mecânica. Por exemplo, no caso de uso de redes sem fio locais, estas possuem um alcance limitado que devem ser avaliadas

pelo jogo. Já redes metropolitanas, como LTE[12], possuem "zonas de sombra" que podem afetar a mecânica e estratégia utilizadas.

- Adaptação do conteúdo de jogo: Conforme cada contexto se apresenta, o jogo pode ajustar seu conteúdo, se adaptando a diferentes ambientes, situações e interações de seus jogadores.

Para cada item listado, Valente et al. [89] apresenta um conjunto de perguntas. Estas estimulam o projetista do jogo a se questionar como cada característica pode ser abordada em sua mecânica.

2.3 Estado dos jogos ubíquos

Jogos ubíquos já são conhecidos desde a virada do século, tempo suficiente para o surgimento de um grande número de títulos que se enquadram nesta categoria. Montola [63] observou que apesar dos avanços alcançados nesta área, ainda existe grande dificuldade destes jogos deixarem de ser apenas "jogos eventos" (*event games*) para se tornarem "jogos como serviços" (*service games*). A grande barreira está em que a maioria dos jogos não evoluem além dos testes de seus protótipos, ou se restringem a eventos de curta duração. Jogos como serviços são o objetivo da indústria, pois envolvem o engajamento de seus usuários por meses a fio.

Uma das primeiras evidências se dá que a maior parte dos jogos publicados na academia restringem seu acesso a locais específicos. Seja pela necessidade de um ambiente controlado, ou pela tecnologia usada em seus protótipos não estarem acessíveis ao público geral. Estes jogos acabam tendo sua avaliação limitada, usualmente não tendo nenhum público ou este se limitando a não mais que uma dezena de pessoas. Neste sentido os jogos como obras de arte têm mais sucesso, chegando a ter centenas de pessoas participando de seus experimentos. Uma das razões está em que estes jogos fogem da abordagem focada na tecnologia, explorando melhor as interações entre as pessoas.

Jogos focados em tecnologia apresentam resultados que não se sustentam quando o efeito "novidade" se esvai. Eles falham em explorar como a estranheza causada pela inovação pode ser um fator motivador constante no jogo ao se tirar proveito da liberdade quanto a ausência de limites no círculo mágico. Estes jogos falham em ousar para aonde se pode evoluir a experiência do usuário. Questões como a imprevisibilidade do ambiente, adaptação de conteúdo e exploração de coincidências são exemplos de espaços onde ainda se precisa inovar nestes jogos.

Sob o viés do *game design*, Stenros et al. [83] questiona que o mesmo foco na tecnologia impede uma maior inovação nestes jogos. Seu ponto é que o projeto de um jogo, apesar de focado em propiciar diversão através do jogar não pode determiná-lo de forma direta. Desta forma, o título apenas define as regras, sendo que o jogar em si é determinado pelo usuário no ato em que interage com o sistema. Neste sentido, os jogos pervasivos ainda tem um considerável espaço para investigação no que diz respeito a como estas regras podem ser exploradas dentro da vasta gama de experiências que os dispositivos ubíquos permitem. Uma proposta é que os títulos devem fugir dos modelos tradicionais

[12]Também conhecida por 4G, vem do inglês "*Long-Term Evolution*", se refere a tecnologia de comunicação móvel em áreas metropolitanas.

de jogos, baseados na visão de ambiente estável e dedicado a esta atividade. Desta forma é possível se experimentar de forma mais efetiva as imprevisibilidades do ambiente ubíquo, permitindo um maior grau de engajamento com os jogadores.

Fica explícito nestas duas visões que, apesar dos avanços encontrados nos jogo ubíquos, ainda existe uma necessidade latente por inovação na sua aplicação. Neste sentido, fugir da visão de um jogo estático para um que incorpore as imprevisibilidades do ambiente dos usuários em sua mecânica se mostra como uma questão ainda a ser explorada.

2.4 Desafios

Um jogo envolve a criação de uma realidade limitada [30], mesmo quando esta tente emular o mundo real. Na concepção deste ambiente é preciso o projeto de diferentes elementos da aplicação. Entre estes incluem-se a forma de jogo e as regras que devem ser respeitadas, a história que traz imersão e a interface a ser utilizada [73]. Diversas variáveis influenciam em como estes elementos podem operar, tais como: o **ambiente** onde o jogo dever ocorrer, sua **flexibilidade a jogadores** e suas interfaces seja **interagindo** com o usuário ou observando o **contexto**.

Com relação ao ambiente, um jogo pode se utilizar de uma faixa que varia de ambientes fechados a totalmente abertos. No primeiro caso, temos maior controle do ambiente. Isto pode ser limitante quanto a liberdade oferecida pelo jogo, porém oferece maior precisão e capacidade de influência dos projetistas do jogo durante a sua execução. Por outro lado, jogos em ambientes abertos são mais difíceis de controlar possuindo a necessidade de se lidar com uma gama maior de incertezas. Isso é compensado pela maior liberdade que estes jogos possibilitam a seus usuários.

A flexibilidade de jogadores é uma característica chave de muitos *ubigames*, sendo que o que mais se espera é que estes permitam uma interação em comunidade. Isso implica em tratar dos desafios que grande parte dos *MMOGs*[13] enfrentam. Apesar disso, diversos jogos ubíquos são focados em grupos mais limitados ou mesmo em um único jogador. Usualmente, esta escolha está ligada a mecânica do jogo ou ao foco da pesquisa motivadora do jogo.

A interação com o usuário é uma das grandes vertentes de pesquisa em jogos ubíquos, seja na pesquisa de novas interfaces ou na utilização de dados de contexto, estas desejam explorar novas formas de se aplicar os conceitos da ubicomp a estes jogos. Dentre estes podemos citar desde exemplos já estabelecidos, como o uso de gestos, projeções e objetos acionáveis até o reconhecimento de emoções e outros dados biométricos.

Como é esperado, as capacidades necessárias para suportar essas características são comuns aos desafios enfrentados por aplicações ubíqua. Dentre estes podemos destacar quatro que são mais evidentes no caso dos jogos ubíquos [31, 72]:

- **Heterogeneidade**: O ambiente inteligente é composto por uma gama variada de plataformas de *hardware* e *software*. Soma-se ainda a existência de uma diversidade de tecnologias de comunicação que convivem no mesmo espaço. Para que se tire proveito do máximo de dispositivos presentes, essas diferenças precisam ser minimizadas, permitindo que a comunicação ocorra entre o maior número de partes possível.

[13] Jogos de Participação Online Massiva, do inglês *Massive Multiplayer Online Games*.

- **Mobilidade**: Tanto usuários como dispositivos podem ir e vir do ambiente conforme desejarem, de forma inesperada ao sistema que os observa. Esta característica deve ser considerada pelos jogos de forma que esta mobilidade possa ser explorada sem prejudicar a experiência dos jogadores.
- **Integração Espontânea**: Conforme novos dispositivos se tornam disponíveis, estes trazem consigo funcionalidades que podem ser exploradas pelos jogos ali presentes. Juntamente, pode se incluir a introdução de novos tipos de dados e recursos que possam ser utilizados.
- **Sensibilidade ao Contexto**: Para que uma aplicação possa reagir às mudanças do ambiente, ela necessita primeiramente ser capaz de percebê-lo. Para tal, informações acerca do *smart space* devem ser disponibilizadas bem como meios de se observar quando estas sofrem alteração.

Os desafios envolvidos não se limitam apenas a estes, porém eles são passos necessários para que os demais possam ser abordados de forma satisfatória. Por exemplo, sem a sensibilidade ao contexto (acesso a informações de contexto), não é possível realizar a gestão do contexto (atuação em decorrência da mudança no ambiente). Por esta razão, estes desafios compõem o foco da pesquisa realizada neste trabalaho. A seguir vemos como eles são abordados por diversas plataformas focadas no desenvolvimento de *ubigames*.

2.5 Plataformas

O desenvolvimento de jogos é uma tarefa complexa. No caso dos jogos ubíquos conta-se ainda com o acréscimo dos desafios anteriormente apresentados. Uma das soluções comuns para facilitar este trabalho é a criação de plataformas ou *middlewares* [72]. O objetivo é abstrair as complexidades dos dispositivos, permitindo o foco na construção dos jogos e reutilizando as soluções para os desafios comuns entre eles.

Tendo em vista a origem na computação ubíqua, alguns jogos se utilizam de plataformas construídas para este tipo de aplicação. Na maior parte, estes jogos são apenas exemplos ou protótipos do que pode ser alcançado com a ferramenta em questão. Dentre as plataformas que exemplificam este caso na academia pode-se citar o *WiFly* [92], plataforma focada no compartilhamento de serviços em redes sem fio, e o *Mobilis*, plataforma para integração de dispositivos móveis [58, 79]. Já entre as plataformas comerciais como os projetos *Spark.io*[14] (com foco na internet das coisas) e *Noam.io*[15] (com foco em integrações locais entre dispositivos), tem-se o caso do jogo CleverPet[16] com o objetivo de treinar e divertir cães.

Apesar do bom resultado apresentado nestes casos, jogos ubíquos apresentam desafios que não são compartilhados por aplicações ubíquas em geral, o que motiva a criação de plataformas específicas para este tipo de aplicação. A seguir são apresentadas algumas das inciativas nesta direção, focadas em jogos ubíquos em geral como em alguns gêneros específicos destes.

[14] `https://www.spark.io`
[15] `www.noam.io`
[16] `http://getcleverpet.com/`

2.5.1 MUPE

O *MUPE* (*Multi-User Publishing Environment*) [56] provê um conjunto de ferramentas para o desenvolvimento de jogos sensíveis a contexto. É fornecido ao desenvolvedor um conjunto de *APIs* para acesso a sete fontes de informações de contexto. Destas se incluem três fontes de dados de localização, informação de clima e esforço físico através de uma bicicleta ergométrica. Os dados destes sensores são fornecidos a um servidor central onde é realizado o processamento das regras do jogo. As informações de saída, fornecida aos jogadores, são providas através de descritores *XML* gerados nesta central. Estes descritores são interpretados por clientes *JME* presentes nos dispositivos móveis (celulares) utilizados pelos jogadores.

Devido a utilização das *APIs*, os desenvolvedores podem utilizar a plataforma na construção da lógica de jogo de forma independente de suas fontes de dados. Por outro lado, além da limitada diversidade de informações fornecidas, os dispositivos que as provêm devem ser registrados anteriormente a sessão de jogo. Desta forma, não é permitida a incorporação de novos artefatos em tempo de execução.

2.5.2 PSD

A plataforma *PSD* (*Player Space Director*) [51] tem por objetivo permitir aos jogos acesso a informações de contexto em tempo real. Isso é alcançado permitindo que diversos sensores e atuadores sejam incorporados a uma sessão de jogo durante a sua execução. Apesar da lógica do jogo ser executada em uma unidade central, esta tem acesso aos dispositivos do ambiente através da injeção de consultas contextuais denominadas *CIQs* (*Contextual Interaction Queries*). Estas consultas observam mudanças no ambiente informando a central quando suas condições são satisfeitas, de maneira assíncrona. Cada dispositivo conectado a central também atua como um "*hub*" de acesso a sensores e atuadores a ele vinculado, expandindo assim a abrangência do sistema.

A utilização das consultas de contexto em conjunto dos "*hubs*" dos dispositivos permite uma grande flexibilidade aos jogos desenvolvidos nesta plataforma. Permitindo que novos artefatos sejam incorporados mesmo que durante a execução de uma sessão de jogo. Porém, a falta de definição de interfaces para a realização do acesso a essas capacidades acaba por minimizar a possibilidade de reuso destes dados bem como a integração de novos tipos dinamicamente.

2.5.3 fAARS

Com foco em jogos de Realidade Aumentada e Alternada, tem-se a plataforma *fAARS* (*for Augmented Alternate Reality Services*) [46]. Esta é uma evolução da ferramenta *fAR-Play* [45], com foco em jogos do tipo "caça ao tesouro". Como um subconjunto dos jogos de localização, convidam o usuário a perseguir objetos e locais baseando-se em sua posição. A plataforma agrega as informações providas por navegadores de realidade aumentada sendo executados nos dispositivos clientes. Desta forma tanto informações de localização para ambientes fechados, utilizando *códigos QR*, como ambientes abertos, utilizando *GPS*, são fornecidas. Com base nisto, o jogo sendo executado em uma unidade central pode tomar suas decisões. Além disso, a plataforma suporta a correspondência de ações do mundo real em ações em um mundo virtual no ambiente *Second Life*.

O uso de clientes locais (nos dispositivos dos usuários) garante que a plataforma tenha acesso a uma diversidade de artefatos. Porém, estes estão limitados a fornecer apenas informações de localização, o que limita a abrangência do contexto que pode ser utilizado. Além do fato de que novos tipos de dados não são passíveis de serem incorporados durante a execução de um jogo, soma-se ainda que as interações móveis são limitadas a mensagens textuais, deixando interações mais complexas ao mundo virtual.

2.5.4 GameWork

O conjunto de ferramentas chamado *GameWork* [82] busca facilitar a tarefa de *game designers* e desenvolvedores na criação de jogos baseados em localização. Uma de suas principais características é permitir que os jogos se adaptem as localidades onde estão executando de acordo com diferentes estratégias pré-selecionadas. Além disto, a plataforma permite a criação de componentes reusáveis entre jogos que podem ser combinados de forma simplificadas por projetistas através de uma ferramenta de autoria. A lógica do jogo é executada em um servidor central e sua interface é composta por páginas *HTML5* interpretadas por navegadores nos dispositivos cliente.

Apesar de limitado a apenas informações de localização, os componentes reusáveis permitem que estas informações sejam expandidas ainda na fase de projeto do jogo. Porém, a falta de interfaces de comunicação e a dependência dos navegadores limita as possibilidades de aquisição de dados da plataforma.

2.6 Comparativo

A relação entre cada uma das plataformas anteriormente apresentadas e os desafios dos jogos ubíquos podem ser observados na Tabela 2.1. O tratamento da heterogeneidade de dispositivos é avaliada por dois critérios: suporte a diferentes redes de comunicação e suporte a diferentes plataformas de *hardware* e *software*. Apesar da predominância do uso de redes de telefonia para grandes distâncias e de *WiFi* para de curto alcance, ainda temos uma grande quantidade de sensores que se utilizam de redes de baixo consumo de energia como *ZigBee* e *Bluetooth*, tornando seu suporte desejado neste tipo de ambiente. Muitas plataformas baseiam sua comunicação em requisições *HTTP* que podem ser utilizadas em diversas tecnologias de rede, porém esta solução possui baixa eficiência de uso da mesma. Com relação as plataformas, temos um uso extenso de clientes locais, em especial navegadores fornecidos por terceiros. Isto é um limitante quanto as possibilidades de interação que podem ser providas bem como os dados que podem ser coletados.

Integração espontânea diz respeito a possuir ferramentas que permitam que aplicações tenham acesso a novos tipos de dados e funcionalidades do ambiente, mesmo em execução. Isto não deve ser confundido com a incorporação de novos dispositivos, que nos casos estudados se limita àqueles que possuem dados compatíveis aos já esperados pela plataforma. A ausência de suporte à integração espontânea é relacionada à deficiência na definição de interfaces de acesso aos dados. Em grande parte isto se dá pela limitação apenas a informações de localização, ou a sua delegando aos desenvolvedores a tarefa de definir e coletar. Apesar de ser desejável que estas interfaces sejam expansíveis, um conjunto básico deve ser disponibilizado a fim de potencializar o reuso de suas implementações entre diferentes jogos.

	Heterogeneidade		Integração Espontânea	Mobilidade	Sensibilidade ao Contexto	Componentes de Jogo
	Múltiplas Redes	Múltiplas Plataformas	Novos Tipos	Novos Dispositivos	Interfaces Definidas	
MUPE	Não	Não[2]	Não	Não	Sim	Não
PSD	Não[1]	Sim	Não	Sim	Não	Não
fAARS	Não[1]	Não[2]	Não	Sim	Não[3]	Não
GameWork	Não[1]	Não[2]	Não	Sim	Não[3]	Não[4]

Tabela 2.1: Plataformas para jogos ubíquos e sua abordagem aos seus desafios.
[1] : Estas plataformas usam requisições *HTTP* como forma de acessar suas funcionalidades, acesso esse que pode ser realizado por diversos tipos de redes.
[2] : Estas aplicações utilizam aplicações clientes, como navegadores, para intermediar suas interações.
[3] : Estas plataformas se limitam a apenas fornecer informações de localização.
[4] : Limitado a apenas itens colecionáveis sem outros componentes esperados de um motor de jogo.

Adicionalmente aos desafios compartilhados com outras aplicações ubíquas, jogos ubíquos também apresentam requisitos encontrados em outros jogos eletrônicos. Sendo assim, existe a necessidade de um suporte por parte de ferramentas que auxiliem não só na construção de seus componentes lógicos como também estéticos (de interação com o usuário). Como pôde ser visto, as plataformas encontradas têm seu foco nos dispositivos e sua interligação, sem fornecer ferramentas específicas a estas necessidades dos jogos. Essa preocupação pode ser encontrada na plataforma *GameWork*, porém ela é restrita a apenas componentes de itens colecionáveis. Isso é limitante, já que a presença de componentes lógicos mais avançados pode permitir o reuso de soluções já estabelecidas como motores gráficos e de física. Em razão desta deficiência, alguns jogos ubíquos se utilizam de motores de jogo genéricos a fim de prover o suporte para estas questões [96].

Há de se observar ainda que a totalidade das plataformas utiliza um arquitetura centralizada para a execução de seus jogos. Esta solução é comum a jogos multijogadores, incluindo jogos massivos, pois simplifica a distribuição dos pontos de tomada de decisão. Porém, em um ambiente ubíquo onde a conectividade pode ser limitada, o acesso de forma *peer-to-peer* pode ser relvante, apesar de ainda não ser explorado por estas plataformas.

Capítulo 3

Jogos Ubíquos Reconfiguráveis

Uma das primeiras formas de entretenimento eletrônico a surgir foram as máquinas de *pinball*, seus componentes consistiam na combinação de circuitos com partes mecânicas. De início estes eram apenas responsáveis pela contabilização de pontos. Porém, com o tempo, suas capacidades lhes permitiram assumir maiores responsabilidades expandindo a diversão com novas possibilidades. Entre estas, a de maior destaque foi permitir que novos desafios fossem conquistados, conforme o jogo desenrolasse. Com a evolução das funcionalidades ao alcance destes circuitos veio o surgimento das primeiras plataformas integralmente eletrônicas, vindo a ser conhecidas como *arcades* ou "fliperamas". Desde então, a evolução tecnológica no âmbito dos jogos eletrônicos vem ocorrendo de forma acelerada. Como exemplares mais notórios temos as diversas gerações de consoles, *PCs* e seus periféricos gráficos e uma imensa variedade de dispositivos móveis. Com uma diversidade tão extensa de plataformas, emerge a questão de como é possível transportar a experiência de um jogo entre elas. Essa capacidade é de interesse da indústria, já que um maior alcance entre as plataformas implica em uma maior abrangência no público atingido por seus títulos. Uma opção para solucionar este problema é desenvolver um título integralmente para cada plataforma, o que é bastante dispendioso. Por esta razão houve o surgimento de ferramentas e tecnologias que permitissem que jogos fossem convertidos para plataformas diferentes, denominados jogos "*Cross-Platform*" ou "Multi-Plataforma".

Apesar do alcance destes títulos ser mais amplo, já estão disponíveis para um maior número de plataformas, o título permanece o mesmo. Ainda que alguns jogos apresentem adaptações para determinadas plataformas, seja para fazer melhor uso de recursos específicos e únicos, ou para satisfazer parcerias comerciais através da oferta de conteúdos exclusivos, a base do jogo permanece inalterada, de forma que a experiência do usuário pouco se diferencia entre plataformas. Em um ambiente ubíquo, onde está presente uma alta diversidade e dinamicidade de dispositivos (bem como suas plataformas), a abordagem multiplataforma se mostra limitada. Para que se possa tirar proveito das capacidades do ambiente, estas devem influenciar os elementos internos do jogo em tempo real. São quatro os elementos [78] que interagem entre si para formar o projeto de um jogo (Figura 3.1):

- A **Estética** estabelece como a percepção de um jogo ocorre[1]. Inclui-se aqui tanto como o jogo percebe as intenções de seus usuários, como estes são estimulados

[1]Apesar do conceito de estética ser amplo, abrangendo significados diversos, neste contexto é limitada apenas a forma como a percepção é materializada (ou adaptada) no jogo.

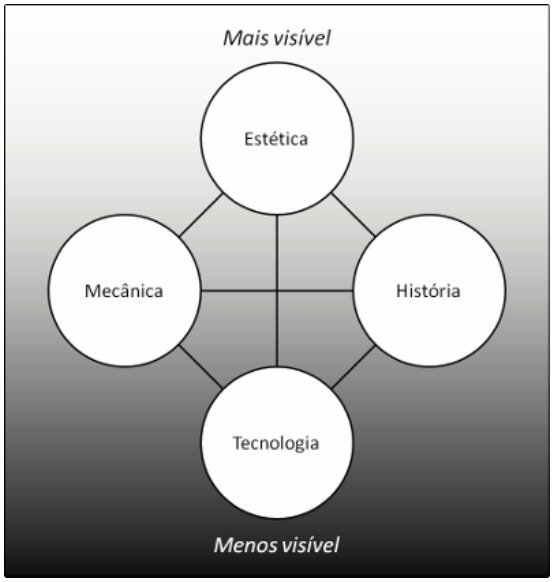


Figura 3.1: Tétrade elementar dos Jogos. [78]

pelo sistema. Pode-se considerar como parte da estética os elemento gráficos, a sonoplastia bem como os controles e outros sensores utilizados.

- A **Mecânica** define o conjunto de regras que estabelecem o funcionamento do mundo criado pelo jogo. É aqui que são definidos os objetos que fazem parte deste universo e como estes são passíveis a interação do usuário. A mecânica também designa as consequências da influência do jogador, criando a sensação de causa e efeito que lhe permite alterar o ambiente virtual vivido.
- A **História** (ou Narrativa) impõe a sequência de fatos ocorridos ao longo do jogo. Ela é fundamental para imergir o jogador no universo projetado. Isso pois ao se definir uma ordem de causalidade na saga vivida, estabelece-se uma consciência da jornada que se deseja criar bem como o papel do usuário.
- A **Tecnologia** é responsável por suportar os demais elementos de forma que estes se concretizem no mundo real. É através dela que o jogo é possível de ser operado, seus estímulos entregues, suas regras computadas e sua narrativa experienciada.

Considerando estes elementos, fica claro que jogos multiplataforma, apesar de adaptados para diversos dispositivos, não modificam a forma como eles funcionam, nem seus elementos. Por outro lado, os jogos ubíquos trazem consigo uma carga considerável de inovação no que tange as tecnologias utilizadas em seus títulos, porém sem muita variabilidade na forma como elas influenciam o projeto do jogo. Observando como as capacidade do ambiente são, ao mesmo tempo, voláteis e diversas, um jogo pode se autoadaptar a elas. É possível reconfigurar seus elementos, proporcionando uma experiência diferente de acordo com cada situação encontrada. Neste contexto são definidos os *Jogos Ubíquos Reconfiguráveis*[2]:

[2]Daqui em diante também referidos apenas por *jogos reconfiguráveis*.

*Um **Jogo Ubíquos Reconfigurável** modifica o comportamento definido em sua Mecânica, Estética, História ou Tecnologia em tempo real, tirando proveito dos recursos e dispositivos presentes no ambiente.*

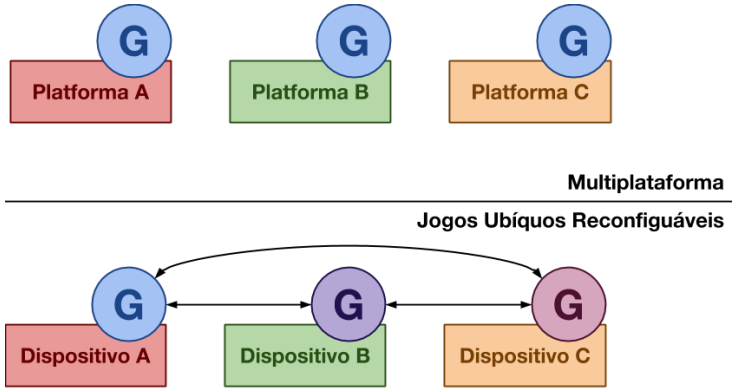


Figura 3.2: Jogos multiplataforma (mesmo jogo em múltiplas plataformas) e jogos ubíquos reconfiguráveis (diferentes adaptações do mesmo jogo que interagem entre múltiplas plataformas).

Jogos reconfiguráveis se diferenciam daqueles definidos como multiplataforma em diversos aspectos (Figura 3.2). Primeiramente, sua adaptação ocorre em tempo de execução e não apenas durante o empacotamento do jogo[3]. Isto permite que ajustes ocorram durante a sessão de jogo. Além disso, existe a comunicação entre os diversos dispositivos do ambiente, fator adicional para ser aproveitado na autoadaptação de seus elementos. Por fim, não só dispositivos podem existir no ambiente como também diferentes instâncias do mesmo jogo. Estas podem interagir entre si, fornecendo a cada jogador uma visão pessoal e distinta de um mesmo universo vivenciado.

Considerando a definição de jogos ubíquos apresentada anteriormente, os jogos ubíquos reconfiguráveis se configuram apenas como um subconjunto destes. Em relação a outras definições eles podem ser caracterizados também dentro dos jogos sensíveis à contexto, já que um de seus objetivos é reagir a alterações no ambiente a sua volta. É importante destacar a forma como isto é proposto nos jogos reconfiguráveis, onde esta adaptação ocorre alterando elementos específicos do jogo conforme detalhado a seguir.

No restante deste capítulo serão apresentados exemplos de jogos ubíquos reconfiguráveis. Seguindo uma classificação escalar, em função dos elementos que são reconfiguráveis. Esta classificação resulta em três níveis distintos de reconfiguração possíveis (Seção 3.1). A seguir, a Seção 3.2 exemplifica dois novos gêneros, criados como parte deste trabalho, de jogos que utilizam dos conceitos de reconfiguração. Por fim, a Seção 3.3 apresenta um catálogo de entradas e saídas que podem ser utilizadas na criação e projeto de títulos deste tipo.

[3]Aqui se incluem diversas técnicas que incluem a tradução, interpretação e máquinas virtuais, pré-complilação e bibliotecas auxiliares. Mesmo assim, todas exigem que o entregável do jogo a ser executado em cada plataforma seja específico para esta.

3.1 Níveis de Reconfiguração

Cada um dos elementos que compõem um jogo influencia aspectos distintos do jogar. A tarefa de ajustá-los de acordo com as modificações que a configuração do ambiente sofre, impacta tanto no projeto do sistema como na experiência vivida pelo usuário. Sendo assim, podemos categorizar a reconfigurabilidade em função de quais destes elementos estão sob influência do ambiente. Três níveis distintos, no que diz respeito ao grau de visibilidade dos elementos aos olhos do jogador, foram observados: Nível 1, Incorporativo; Nível 2, Adaptativo; e Nível 3, Complementar.

3.1.1 Nível 1 - Incorporando Dispositivos

Neste Nível, o objetivo da reconfiguração está limitado a observar os dispositivos disponíveis e incorporá-los ao jogo. Isto restringe a adaptação afetando apenas a tecnologia e mantendo os demais elementos (mecânica, estética e história) inalterados. Usualmente isto é alcançado pela integração de novos meios de entrada e saída, equivalentes àqueles já projetados para interagir com o jogo. Com relação aos dispositivos de entrada, é comum a adição de novos controles sob o comando de novos jogadores, ou a apresentação de novas opções ao usuário corrente. A incorporação de dispositivos de saída pode ser encontrada no suporte fornecido pela maior parte dos consoles de quarta geração. Nestes existem ferramentas que possibilitam a transmissão do jogo para dispositivos portáteis, estendendo a sessão de jogo mesmo quando se está distante da televisão.

Figura 3.3: *Battefield 4™: Commander* sendo jogado em um *tablet*. [7]

A incorporação de dispositivos de entrada pode ser vista em títulos comerciais, como "*Super Sync Sports*" [4], ou da academia, como "*uMoleHunt*" [23]. Estes demonstram como novos controles podem ser obtidos de forma dinâmica incorporando-se dispositivos móveis. Outro exemplo deste tipo de estratégia é o jogo "*Battlefield 4™* " (Figura 3.3). Conhecido por ser um jogo de tiro em primeira pessoa (*FPS*[5]), este possui um modo de estratégia em tempo real (*RTS*[6] denominado de "*Commander*". Este modo está disponível tanto para jogadores utilizando um console como um *tablet*[7]. O recurso "*Remote Play*" disponível no console "*Play Station 4™*" é um exemplo de como saídas podem ser incorporadas

[4] `http://chrome.com/supersyncsports/`
[5] Do inglês *First Person Shooter*.
[6] Do inglês *Real Time Strategy*.
[7] `https://play.google.com/store/apps/details?id=com.ea.game.warsawcommander_row`

aos títulos. Este recurso permite a jogos como "*God of War*" [8] (Figura 3.4) e "*Bionic Commando*"[9] serem experienciados tanto na tela da TV como em dispositivos móveis. Este recurso garante que a transição entre tais formas de interação seja transparente ao jogador, assegurando a continuidade da sessão corrente.

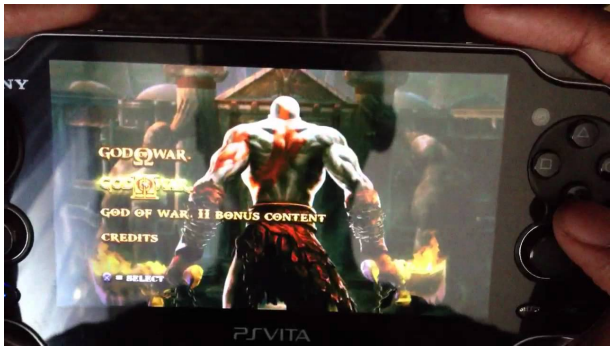


Figura 3.4: Jogo *God of War* rodando em um dispositivo *PS-Vita* usando a tecnologia *Remote Play*. [8]

3.1.2 Nível 2 - Adaptando a dispositivos

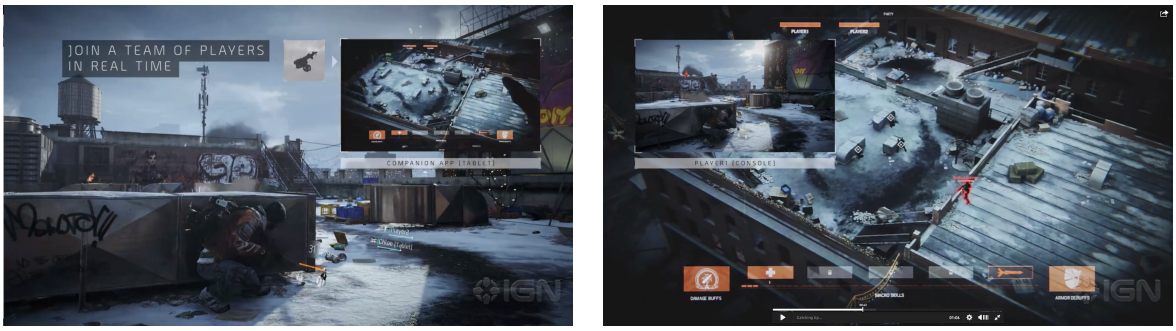
A reconfiguração de jogos no Nível 2 não se limita apenas à adaptação da tecnologia, sendo aqui também adaptados os elementos de mecânica e estética de acordo com as capacidades oferecidas por cada dispositivo. Ainda que a história permaneça inalterada, este tipo de reconfiguração permite mudanças que disponibilizem tanto novas mecânicas, de acordo com os recursos presentes, ou apenas a adição de estímulos complementares à experiência.

Figura 3.5: Informação complementar de *GPS* sendo exibida no jogo *Forza Horizon* utilizando o suporte do *Xbox SmartGlass*. [10]

O jogo de corrida "*Forza Horizon*"[10] (Figura 3.5) faz uso da tecnologia "*Xbox SmartGlass*" para encontrar dispositivos móveis presentes no ambiente. Fazendo uso de suas

[8] `https://godofwar.playstation.com`
[9] `http://www.bioniccommando.com/`
[10] `http://www.forzamotorsport.net/`

telas, ele provê informações adicionais ao jogador que não estariam disponíveis caso contrário. No caso, é exibido na tela móvel um mapa representativo do mundo do jogo, onde é possível visualizar os locais nos quais determinados eventos estão ocorrendo. Isso permite que o jogador, mesmo com seu foco na TV, observe de forma periférica esta informação, ou mesmo receba auxílio de outras pessoas que o estejam acompanhando.



(a) Modo *FPS* disponível no console.

(b) Modo *RTS* disponível no *tablet*.

Figura 3.6: Dois experiencias distintas do jogo *The Division*. [11]

No caso do título "*The Division*"[11] (Figura 3.6), a presença de dispositivos com capacidades diferentes é utilizada para propiciar experiências distintas. Observando o que cada aparelho tem a oferecer são disponibilizadas, em um mesmo universo, mecânicas distintas ao jogador. Caso este esteja utilizando um console, tem então a sua disposição um modo de tiro em primeira pessoa (*FPS*). Já aqueles que estejam fazendo uso de um *tablet* experimentam um modo de estratégia em tempo real (*RTS*), mais adequado para a interação em uma tela de toque.

3.1.3 Nível 3 - Complementando a experiência

Modificar todos os elementos de um jogo, incluindo sua narrativa, permite que usuários acessem, ou mesmo criem, novas histórias. Tirando proveito das capacidades presentes em cada dispositivo, bem como a configuração do ambiente, é possível ajustar a trama principal do título em execução. De acordo com cada situação, subtramas podem ser adicionadas, removidas ou até modificadas. Isto expande a experiência do jogo, fazendo com que cada artefato presente acrescente novas possibilidades de interação a aventura, tornando-a distinta das demais. Outra forma de tirar proveito desta liberdade é através da cooperação entre instâncias diferentes de jogos. Estas podem cooperar entre si, interferindo nos resultados alcançados em cada uma delas.

Um exemplo de como a presença de dispositivos distintos é aplicada para influenciar os caminhos da narrativa, pode ser encontrada no título "*Dead Rising 3*"[12] (Figura 3.7). Este jogo permite que o usuário sincronize seu *smartphone* com o console, passando a representar o dispositivo pertencente ao personagem principal. Durante a sessão, o telefone é utilizado por outros personagens para convidar o usuário a tomar parte em missões

[11] `http://tomclancy-thedivision.ubi.com/`

[12] `https://store.xbox.com/pt-BR/Xbox-One/Games/Dead-Rising-3/affdf371-f512-4d8d-a9e9-dac4524a0e3a`

especiais. Desta forma, novas tramas são habilitadas, sendo estas não acessíveis aqueles que não façam uso deste recurso.

Figura 3.7: Tipos distintos de interação disponíveis ao jogador utilizando o seu *smartphone* conectado ao jogo *Dead Rising 3* através da tecnologia *Xbox SmartGlass*. [12]

A iniciativa "*Pokemon Dream World*"[13] (Figura 3.8) desenvolvida pela *Nintendo* visa integrar diversos títulos diferentes da franquia *Pokemon* em uma experiência conjunta. Seu título principal disponível ao console portátil "*Nintendo DS*" é chamado de "*Pokemon Black and White*". Este convida seus jogadores a capturar e treinar monstros conhecidos como *pokemons*. Seja cuidando destes ou utilizando eles em batalhas contra outros treinadores. Em um navegador *web* se tem acesso a um conjunto de jogos casuais que permitem aumentar o grau de amizade ou habilidade que os monstros possuem junto a seu dono. O resultado destes jogos se reflete no título principal, expandindo as consequências das ações e conectando suas narrativas. De forma complementar, o dispositivo "*Pokewalker*" permite acrescentar interações distintas ao jogador permitindo que este leve seu *pokemon* para um passeio. Artefato este que opera como um pedômetro, onde quanto mais longe se vai, mais experiência é acumulada pelo monstro selecionado. Soma-se ainda que o dispositivo permite a interação com outros treinadores encontrados pelo caminho.

3.2 Gêneros de jogos reconfiguráveis

Jogos como *Treasure* [43] expandem as possibilidades narrativas de um jogo. Para isso, o usuário deve pré-configurar os objetos do ambiente a história desejada, criando inúmeras variações desta experiência. Jogos reconfiguráveis levam isso a um patamar além, onde estas alterações ocorrem em tempo de execução. Isto possibilita que novos tipos e gêneros de jogos, que tirem proveito destas características, sejam criados. Aqui são apresentadas duas novas possibilidades de gêneros de jogos ubíquos reconfiguráveis.: Jogos Espontâneos e Jogos Abertos.

Jogos Espontâneos possuem a habilidade de se autoadaptar a situações não planejadas. Seu objetivo está em criar experiências que sejam ativadas em qualquer lugar,

[13] `http://www.pokemon-gl.com/`

(a) No console portátil *Nintendo DS*.

(b) Em um navegador *web*.

(c) No dispositivo *Pokewalker*.

Figura 3.8: Três distintas experiências do jogo *Pokemon Black and White* na plataforma *Pokemon Dream World*. [13]

integrando os elementos presentes no jogo. Os **Jogos Abertos**, por sua vez, têm como propósito permitir que tanto desenvolvedores como jogadores modifiquem sua experiência conforme desejar. Desta forma, novos artefatos e recursos podem ser utilizados, ou até mesmo comportamentos modificados, de forma não previstas pelos projetistas do jogo. Não sendo excludentes, estes gêneros são também passíveis de serem mesclados entre si, explorando ambas as liberdades por eles propostas e criando uma esperiência mais dinâmica aos seus jogadores. Como demonstrado no jogo *Treasure* [43], a liberdade de correlacionar elementos de um jogo amplia o nível de interação de seus jogadores. Nestes dois gêneros esta personalização é ainda mais podencializada, não só por exemplificar casos em ambientes abertos, mas por explorar outros elementos de personalização.

3.2.1 Jogos Espontâneos

Jogos Espontâneos observam o ambiente ao redor do usuário e avaliam quais de seus elementos podem ser incorporados como parte da sessão corrente. Tais artefatos não se limitam apenas à absorção de novos dispositivos, sensores ou atuadores, mas também outras sessões de jogo que lá estejam presentes. Indo além de apenas incorporar estes, como também se ajustar quando sejam removidos do ambiente. Isso deve ser considerado, já que em um ambiente ubíquo a variabilidade de seus elementos não é previsível. Dessa forma cada configuração de ambiente se torna uma experiência completamente diferente do jogo. Característica esta que acaba por funcionar como um incentivo a seus jogadores buscarem novos espaços para encontrar aventuras diferentes. A seguir é apresentado o jogo espontâneo *uClue*, como exemplo de como o gênero pode ser aplicado.

uClue

O jogo *uClue* é uma variação do jogo conhecido como "Detetive"[14]. Neste jogo, o usuário deve resolver o mistério que cerca um assassinato. A história começa com cada jogador

[14]Também conhecido como *Clue* ou *Cluedo*, projetado por Anthony E. Pratt e publicado por Waddingtons, Parker Brothers, Hasbro e Estrela

sendo requisitado pela senhorita *Scarlet*[15]. Quando o número desejado de jogadores se junta à sessão, respondendo ao chamado, um blecaute ocorre. Assim que as luzes retornam, a senhorita *Scarlet* é encontrada morta. Agora, cada um dos jogadores deve descobrir onde, como e por quem o crime foi cometido.

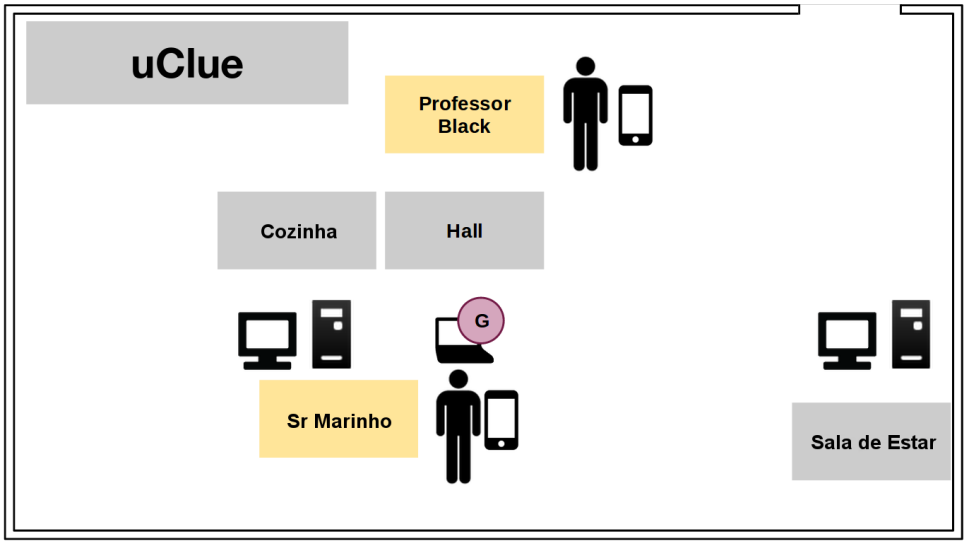


Figura 3.9: Ambiente exemplo do jogo *uClue* e as metáforas para os dispositivos ali presentes.

Se adaptando ao ambiente, o jogo observa os dispositivos disponíveis incorporando-os como elementos jogáveis. Computadores pessoais e *laptops* são utilizados para representar os cômodos da mansão da senhorita *Scarlet*. Celulares e outros dispositivos portáteis representam os personagens (detetives) personificados pelos jogadores (Figura 3.9). Quando um som é reproduzido em um dos cômodos, os jogadores devem correr para a frente da tela deste descobrindo o que lá ocorreu. Em cada evento são reveladas pistas acerca do que ocorreu naquele local, auxiliando na investigação do mistério. Para que esta informação seja recebida pelo jogador, este deve chegar até o cômodo dentro de um tempo estabelecido. O primeiro detetive a desvendar o mistério corretamente, informando onde o crime ocorreu, quem o realizou e qual a arma foi utilizada, ganha o jogo.

Conforme novos dispositivos se tornam disponíveis o jogo se modifica para utilizá-los como parte da história ou mecânica. Por exemplo, caso um medidor de batimentos cardíaco esteja disponível, este é utilizado para aumentar ou diminuir a frequência de eventos (e o tempo requerido para se chegar a um cômodo). Novos computadores são incorporados como novos cômodos que são descobertos, enquanto novos celulares são novos personagens que lá se encontram (Figura 3.10). Caso uma nova sessão de jogo seja encontrada, ambas podem ser associadas de forma que os jogadores de cada uma possam interagir entre si. Assim, um mistério com múltiplos crimes passa a ser resolvido. Esta adaptação faz com que cada configuração de ambiente seja uma nova experiência de jogo, distinta tanto em complexidade como em dificuldade, variando apenas os elementos presentes.

[15]Os nomes de quem requisita ajuda é alterado para cada sessão de jogo. Desta forma cada um destes possui um mistério diferente a ser resolvido.

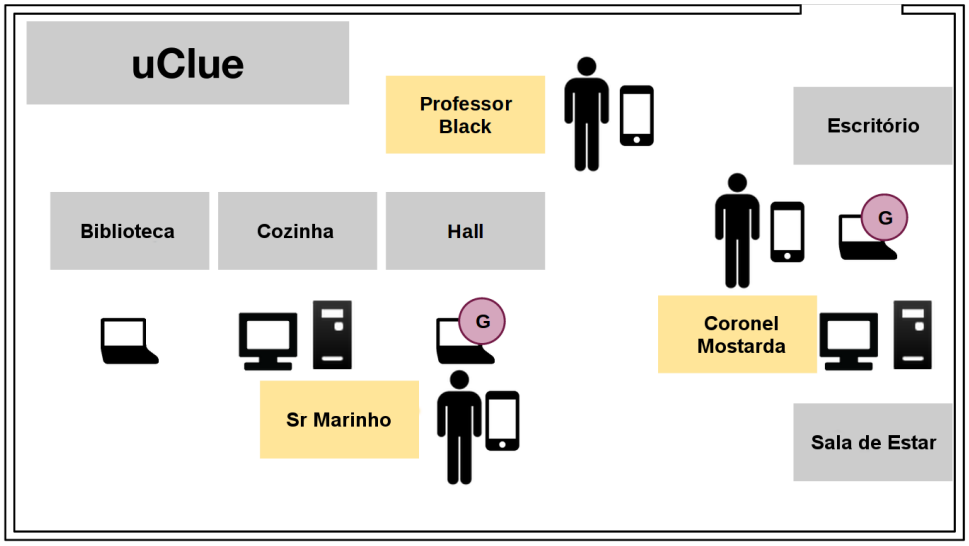


Figura 3.10: Ambiente exemplo do jogo *uClue* com a chegada de novos dispositivos e múltiplas instâncias de jogo.

3.2.2 Jogos Abertos

Jogos eletrônicos de sucesso tendem a ter sua continuidade expandida além do título original. Sem a necessidade de recorrer a criação de continuações, que em si são caracterizadas como novos títulos, existem também as expansões e *DLCs*[16]. Estes ampliam a história e a mecânica de seus títulos originais, usando a mesma base tecnológica, prorrogando sua vida útil. Porém, estas formas de modificar os títulos ocorre de forma planejada, sob o controle de seus criadores e da equipe que deu origem ao título. Modificações de jogo (ou apenas *MODs*) são ferramentas que permitem que usuários alterem os elementos do mesmo, de acordo com a sua criatividade [77]. Isto tem permitido a evolução e diversificação de títulos através de conteúdo criado por seus próprios usuários. Característica essa que de tão benéfica tem sido fortemente estimulada pela indústria, já que aumenta não só o engajamento como o sentimento de comunidade. De forma similar, os Jogos Abertos tem por intuito permitir que os usuários possam alterar os elementos e comportamentos de um jogo ubíquo. Porém como jogos reconfiguráveis, estas modificações são realizadas em tempo de execução, de maneira transparente ao jogador ou mesmo como parte da mecânica. Desta forma não apenas desenvolvedores podem expandir a experiência do jogo mas também usuários comuns. Essa característica favorece a criação de *designs* emergentes [38] onde o jogador tem grande liberdade de criar em cima da experiência do jogo, alanvancando sua experimentação e assim longevidade.

Para alcançar este propósito, dois elementos do jogo devem ser flexíveis a mudanças: a tecnologia e a mecânica. No que tange a tecnologia, entradas e saídas devem ser maleáveis de maneira que seja possível comutar estes com outros de uso equivalente. Já com relação a mecânica do jogo, esta deve permitir que partes do seu comportamento sejam alteradas possibilitando a criação e inserção de novas interações e reações.

Em um jogo aberto, está presente a escolha de como suas entradas e saídas podem ser usadas, o que altera de forma significativa a experiência do usuário. Por exemplo, um

[16]Do inglês *Downloadable content* ou Conteúdo Adquirido Digitalmente.

jogo de plataforma pode ser desenvolvido para que sua interação seja realizada através de um controle de jogo tradicional. Porém, um jogador mais ávido pode escolher por utilizar um sensor de movimento para chegar ao mesmo propósito. Dando a este usuário um sentimento de maior imersão com relação às ações de seu personagem. Considerando que um controle é composto por um conjunto de vetores de movimento (comandos direcionais) e botões de controle (comandos de ação), o mapeamento destas funções para um conjunto de movimentos é factível. Neste caso, se a mesma interface for respeitada por ambos os dispositivos, a troca de um pelo outro pode ocorrer sem qualquer alteração ao jogo em si.

Por outro lado, a modificação do comportamento de um jogo requer a habilidade de injetar informação de execução nos componentes responsáveis por suas capacidades. Isto pode ser alcançado utilizando-se de técnicas de mobilidade de código [24]. Isso permite que o jogador aprimore o jogo modificando partes do mesmo, seja por gosto próprio ou objetivando aprimorar sua técnica. Por exemplo, em um jogo de estratégia, onde o jogador comanda um conjunto de unidades, a modificação de comportamento pode expandir consideravelmente as possibilidades de elaboração de seus planos. No caso de uma unidade que tenha um comportamento padrão como "bater e correr", esta pode ser modificada para apenas atacar oponentes mais fracos (uma natureza covarde). Desta forma, o jogador pode ajustar como cada unidade opera para melhor se adequar a sua tática.

No Capítulo 5 são apresentados exemplos de jogos reconfiguráveis, em especial o jogo aberto *uSect* que faz tanto uso da flexibilidade em sua tecnologia como em sua mecânica.

3.3 Meios de Interação

Para que seja possível reconfigurar os jogos de acordo com o ambiente é necessário que se conheça as capacidades que este tem a oferecer. Por mais que estes títulos sejam passíveis de se adaptar de acordo com o usuário e com o que existe a sua volta, ainda cabe ao *game designer* a tarefa de elaborar as metáforas do jogo. São estas que permitirão que o jogador reconheça no mundo real os elementos criados no universo virtual de forma natural. Além disto, a integração espontânea de dispositivos e a sensibilidade ao contexto dependem de se possuir um conhecimento prévio de quais informações e interações são possíveis. Sem esta base de conhecimento não é possível que os elementos do ambiente sejam propriamente utilizados, nem tanto que novos sejam incorporados. Por esta razão, aqui são listados diversas entradas e saídas passíveis de aplicação em jogos ubíquos. Juntamente a estas são apresentados também exemplos de dispositivos e métodos de como podem se chegar a estas formas de interação. Esta lista toma por base o estudo de mais de 80 títulos (vide apêndice A) contendo jogos ubíquos tanto da academia como da indústria.

- **Usuário**: Informações de usuário são tão importantes quanto complexas, como pode ser exemplificado pela ontologia *GUMO* [48] e seus quase 1.000 conceitos relacionados a este tipo de dado. Entre estes, a identificação do usuário pode ser considerada como o de maior importância para a construção de um jogo. Perceber a presença e conhecer quem são as pessoas em um determinado local pode ser realizado através de diversos meios. Desde soluções simples, como relacionar uma pessoa com a presença de seu dispositivo pessoal (seu celular) ou etiquetas sem fio (como *RFID*), a mais complexas como reconhecimento de face e padrões de voz.

- **Controle**: Controles são a forma mais tradicional de interagir com um jogo. Sua variabilidade é grande, desde o simples *joystick* do console *Atari* ao complexo *PS4™DualShock®4*. Apesar disso, sua composição pode ser vista como um grupo de alavancas, botões e gatilhos. Além dos controles em si, a mesma função é obtida por meios como *mouses*, teclados, sensores de movimento e até sensores presentes em *smart phones* como acelerômetros e telas de toque.

- **Objeto**: Assim como saber quem está em determinado local é uma informação importante, o mesmo se aplica a outros objetos lá presentes. Conhecer quais artefatos encontram-se no ambiente permite ao jogo incorporá-los a narrativa, aumentando o nível de relacionamento entre o mundo real e o virtual. A forma mais comum de obter esta informação é através de etiquetas sem fio (como *RFID* e *NFC*), especialmente por serem soluções de baixo custo. Porém, o uso de etiquetas visuais (como códigos *QR*) e reconhecimento de imagens também é uma possibilidade.

- **Interação com Objeto**: Além de reconhecer a presença de um objeto, a interação com eles permite um maior grau de imersão com o ambiente. Desta forma, pode-se não apenas saber que um baú do tesouro se encontra na sala, mas também identificar que ele foi aberto. Este tipo de interação pode ser percebida por diversos meios: botões podem ser usados para se obter interações diretas; sensores magnéticos podem assinalar que algo foi aberto ou dobrado; acelerômetros indicam movimento ou gestos.

- **Posição Relativa**: Posicionamento é um dado crucial na criação de jogos onde o movimento dos jogadores é parte integral da mecânica. A posição relativa indica onde um objeto, ou alguém, está em relação a um marco conhecido. Esta restrição se mostra interessante quando este marco e o alcance de sua percepção possui relação com o jogo, como uma sala ou prédio. Um bom exemplo de sensor que provê esta informação é o *Kinect* que provê a posição em três eixos de objetos em seu campo de visão, bem como seu rastreamento. Outras forma de obter este dado é através da triangulação de sinais de rádio, onde as antenas são utilizadas como os marcos.

- **Posição Absoluta**: Posicionamento absoluto envolve conhecer as coordenadas de um objeto ou pessoa no globo terrestre. Esta informação é fundamental em jogo em ambientes abertos ou que usam áreas metropolitanas ou globais. Estes jogos acabam requerendo pouco esforço de configuração para determinar suas fronteiras, utilizando este dado. A forma mais comum de se obtê-lo é através de dispositivos de *GPS*, porém o mesmo dado pode ser calculado usando-se um marco conhecido e sua posição relativa a este.

- **Armazenamento**: Armazenar informações nos dispositivos permite ao jogo distribuir sua lógica e criar interações espontâneas sem a necessidade de um servidor central. Por exemplo, um jogo pode permitir que seus usuários deixem uma marca em objetos do ambiente armazenando sua identificação nos mesmos. Desta forma, outros jogadores podem interagir com aquele objeto e obter o rastro de seus companheiros.

- **Entrada de Áudio**: Microfones provêm tanto informações pessoais como públicas de áudio que podem ser utilizadas no jogo. Individualmente estas podem ser usadas

para se obter comandos ou mensagens, porém quando agrupadas podem interligar ambientes não fisicamente conectados através do seu som ambiente.

- **Saída de Áudio**: Caixas de som públicas, ou fones de ouvido, permitem que jogadores tenham estímulos auditivos tanto públicos como privados. Os efeitos projetados pelo som podem informar sobre ações que ocorrem no jogo, como explosões, conquistas ou ataques. A música de fundo atua no grau de imersão do jogo e influencia nas emoções e cadência do jogar.
- **Captura de Imagem**: Imagens do mundo real normalmente são capturadas por câmeras e podem ser utilizadas para integrar o que o usuário vê e o que o jogo percebe. Exemplos podem ir desde a simples aplicação de texturas nos cenários a coleta de objetos em um jogo de "caça ao tesouro".
- **Reprodutores de Vídeo**: O estímulo visual é o mais utilizado nos jogos eletrônicos. Telas podem ser encontradas em monitores, *LPDs* [17], telefones ou mesmo embarcados em objetos do ambiente. O foco em retornos visuais pelo sistema é util mas deve ser usado de forma cautelosa para não dispersar a atenção do usuário, reduzindo seu engajamento.
- **Entrada de Texto**: Comandos textuais representam um tipo mais geral de entradas providas pelo usuário ao jogo. Apesar deste tipo de interação estar usualmente ligada ao uso de teclados, pode-se também utilizar reconhecimento de voz e gestos. Por ser uma comunicação mais aberta, esta demanda respostas mais inteligentes por parte dos jogos, o que transmite um maior grau de realidade.
- **Saída de Texto**: Apesar do retorno visual ser o de maior destaque nos jogos eletrônicos, saídas de texto são tecnicamente mais fáceis de se realizar como, por exemplo, utilizando mostradores de *LED* de baixo custo. Esta limitação pode ser explorada em jogos onde pequenas informações textuais possam incrementar objetos do mundo real.

É válido ressaltar que esta lista não visa exaurir por completo todos os tipos de entradas e saídas para uso presente, ou futuro, em jogos ubíquos. Pode se citar exemplos que aqui não se encontram, como as informações biométricas. Dentre poderiam ser incluídos batimentos cardíacos, condutividade da pele e emoções encontradas em diversos títulos. Apesar destas, e outras informações, serem de grande interesse para os jogos ubíquos, elas consistem em agrupamentos ainda muito específicos. Desta forma, espera-se que esta lista venha a ser atualizada e expandida conforme novos tipos tomem cena neste jogos. Apesar destas limitações, a informação aqui presente é de relevantes a *game designers* na formulação de metáforas a serem exploradas em seus jogos.

[17] Do inglês *Large Public Displays* ou Grandes Telas Públicas.

Capítulo 4

Plataforma *uOS* para Jogos Ubíquos

Jogos Ubíquos compartilham os desafios presentes tanto em aplicações da *ubicomp* como em jogos eletrônicos. Questões como heterogeneidade e mobilidade se somam a outras como aprimoramento da renderização gráfica e a física de objetos virtuais. Quando o que se objetiva é um maior grau de imersão, este conjunto maior de fatores deve ser tratado com maior importância. Dessa forma, os benefícios providos ao se utilizar tecnologias ubíquas não devem ocorrer em detrimento das funcionalidades esperadas por um jogo eletrônico. Observando as diversas iniciativas existentes com o objetivo de auxiliar na construção de jogos ubíquos, nenhuma endereça de maneira abrangente os principais desafios envolvidos: heterogeneidade, mobilidade, integração espontânea e sensibilidade ao contexto. Os mesmos desafios são compartilhados no desenvolvimento dos - aqui propostos - jogos ubíquos reconfiguráveis.

Dentre as plataformas encontradas, nota-se também a ausência de suporte à construção de jogos, de forma similar àquele provido por motores de jogo[1]. Estes consistem em camadas de *software* que auxiliam no desenvolvimento de títulos ou famílias de jogos. O uso desses motores diminui de forma significativa o esforço desprendido na implementação, promovendo o reuso de soluções através do emprego de componentes lógicos como cenas e mapas. Através de *APIs* disponibilizadas, tem-se acesso a recursos avançados como a renderização de gráficos, simulação de física e o processamento de entradas [71]. Algumas das ferramentas existentes incluem também um suporte completo ao desenvolvimento, permitindo o controle e criação de conteúdo e comportamentos através de ferramentas de autoria gráficas[2].

Dentre as vantagens de se utilizar um motor de jogo, se destaca a flexibilidade dada à criação de títulos pertencentes a diversos tipos e gêneros distintos. Isso é possível pois, ao se abstrair os desafios comuns entre eles, permite-se aos desenvolvedores maior foco no desenvolvimento e experimentações de suas aplicações. Somam-se aos seus benefícios a disponibilização de componentes lógicos que acabam por estabelecer um conjunto de boas práticas bastante difundidas entre os profissionais da área, consolidando uma base de conhecimento compartilhada. Isso minimiza a curva de aprendizado e facilita a comunicação entre times distintos, promovendo o reaproveitamento de código bem como trocas de experiências [41]. O impacto da aplicação deste tipo de ferramenta no mercado

[1] Do inglês *Game Engines*.

[2] Como exemplo, podemos citar a plataforma *Unreal* `http://www.unrealengine.com/`

é evidente quando na atualidade a grande maioria dos jogos é desenvolvido utilizando um motor ou envolve a construção de um que dê suporte a suas necessidades específicas.

Figura 4.1: Plataforma *uOS* para jogos ubíquos.

Fica claro, então, que além do suporte provido por plataformas voltadas a aplicações ubíquas, jogos ubíquos necessitam também de apoio especializado no desenvolvimento de *games*. A plataforma aqui apresentada tem o intuito de nivelar essas dificuldades, favorecendo a criação deste tipo de *software*. A Figura 4.1 representa os componentes que compõem a plataforma, sendo esta construída com o apoio do *middleware uOS*. Este é descrito em mais detalhes na Seção 4.1. Sua função consiste em prover a descoberta e comunicação entre dispositivos no ambiente.

Compondo a plataforma, tem-se ainda o motor de jogo *uImpala* (Seção 4.2) que simplifica o esforço de desenvolvimento de novos títulos. Este oferece um conjunto de abstrações e *APIs* de apoio à implementação de títulos ubíquos seguindo padrões estabelecidos no mercado e academia, o que diminui a curva de aprendizado bem como facilita a incorporação de ferramentas e bibliotecas auxiliares. Um *plugin* de comunicação para o motor comercial *Unity 3D* também está disponível. Desta forma, jogos que se utilizem deste motor também podem acessar, e disponibilizar, recursos no ambiente (Seção 4.3). Com a finalidade de oferecer suporte a mobilidade de código, foi desenvolvido o *Execution Driver* apresentado na Seção 4.4, sendo este de grande valia na construção de jogos abertos. As alterações realizadas no *middleware* durante a realização desta pesquisa são apresentadas na Seção 4.5 .Por fim, a Seção 4.6 apresenta a definição de uma hierarquia de recursos e suas interfaces, facilitando o reuso destas no desenvolvimento entre títulos distintos.

4.1 uOS Middleware

O *middleware uOS*[3] implementa a arquitetura *DSOA*[4] [22] com foco na comunicação entre dispositivos. Apresenta-se a seguir um conjunto de conceitos e requisitos relacionados à criação de ambientes inteligentes, graficamente ilustados na Figura 4.2:

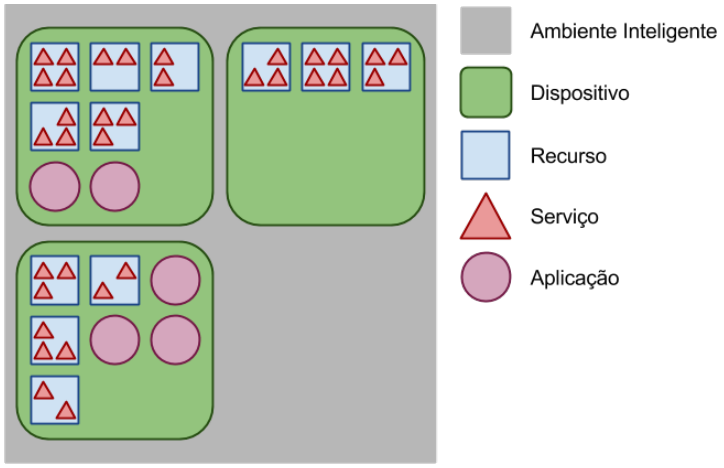


Figura 4.2: Ambiente Inteligente definido de acordo com a arquitetura *DSOA*.

- O **Ambiente Inteligente** é um conjunto não vazio nem unitário de dispositivos providos de poder computacional conectados através de uma rede de comunicação de forma colaborativa.
- Um **Dispositivo** é um artefato computacional com a capacidade de se comunicar, bem como abrigar aplicações ou disponibilizar recursos ao ambiente inteligente.
- Um **Recurso** define-se por um grupo de funcionalidades logicamente relacionadas e acessíveis ao ambiente por meio de interfaces definidas.
- Um **Serviço** consiste na implementação de uma funcionalidade disponível ao ambiente utilizando-se de um recurso e sua interface pública conhecida.
- Uma **Aplicação** implementa um conjunto de regras e comportamentos relacionados aos recursos e usuários presentes no ambiente inteligente.

Como exemplo de um ambiente inteligente tomemos uma sala de estar onde se encontra disponível uma rede de comunicação sem fio através de um ponto de acesso *WiFi*. Ali estão presentes diversos dispositivos dotados de poder computacional tais como: uma televisão, um *smartphone*, um *tablet* e um ar condicionado. Dentre todos, o ar-condicionado não pode ser considerado como parte do ambiente inteligente, já que lhe falta a capacidade de se comunicar com os demais dispositivos. Observando o *smartphone* podemos reconhecer neste uma diversidade de recursos que podem ser utilizados pelo ambiente, como sua tela, câmera, armazenamento e vários sensores dentre os quais podem ser citados os de movimento e temperatura, entre diversos outros. Tomando o recurso de câmera por

[3] `https://github.com/UnBiquitous/uos_core`
[4] Do inglês *Device Service Oriented Architecture* ou Arquitetura Orientada a Serviços para Dispositivos.

exemplo, este pode oferecer serviços de transmissão de vídeo, captura de imagem, nível de luminosidade ou mesmo detecção de movimento. Uma aplicação de notícias sendo executada no *tablet* pode estar atenta à presença do usuário, obtendo sua identificação quando o *smartphone* for identificado no local. Quando isto ocorrer, a aplicação busca a interface mais adequada, como a televisão, e a utiliza para a exibir as notícias de interesse da pessoa reconhecida.

A *DSOA* estabelece um conjunto de requisitos que devem ser providos pela plataforma que a implementa como os serviços e recursos são operacionalizados. Estes são divididos em três categorias: transporte de dados, modelo de comunicação e equivalência de recursos.

No que tange ao transporte de dados durante a comunicação de um serviço, este pode ocorrer sob duas formas distintas. *Mensagens Discretas* permitem que sejam transportadas informações textuais onde as partes conhecem seus delimitadores. Este tipo de comunicação é adequado para a execução de comandos ou realização de consultas. Já uma *Comunicação Contínua* se caracteriza pela existência de um fluxo de dados binários entre as partes, não havendo a necessidade de acordo com relação ao seu término. Esta forma de comunicação é mais apropriada para a transmissão de arquivos ou *streaming* de vídeo/áudio.

O modelo de comunicação entre dispositivos pode ser realizado de forma tanto síncrona como assíncrona. Uma comunicação *síncrona* se inicia com a criação de uma requisição por parte do consumidor do serviço. Em seguida, este deve aguardar a resposta a ser produzida pelo provedor escolhido. Já um modelo *assíncrono* envolve o uso de eventos. Desta forma, o consumidor se registra junto a um provedor para o recebimento de notificações quando estas venham a ocorrer, caso ocorram. Enquanto o modelo síncrono é mais apropriado para serviços que representem comandos e consultas, o assíncrono se mostra de maior valia ao se observar alterações no ambiente ou interações do usuário.

No ambiente, um ou mais recursos podem possuir serviços similares ou mesmo equivalentes dentro de um contexto. Tomando por exemplo um jogo com o qual se interage apenas com o uso de dois botões de ação. Este pode utilizar como dispositivo de entrada tanto um *mouse* como um controle. A *DSOA* estabelece que todo recurso é passível de definir sua cadeia de equivalência junto a outros recursos. Neste caso, é necessário que os serviços do recurso ao qual se declara equivalência sejam também implementados, funcionando de forma similar ao que ocorre no caso de herança em um modelo orientado a objetos.

O *middleware uOS* é a implementação de referência da *DSOA*, fornecendo em sua completude funcionalidades que satisfazem os requisitos estabelecidos. Isto permite a integração e troca de mensagens entre dispositivos heterogêneos. Para isto é utilizado como formato de comunicação o conjunto de protocolos leves *uP* [21]. Estes adotam como base para suas mensagens o formato *JSON*[5], assegurando que a coordenação de serviços ocorra de acordo com o definido pela arquitetura enquanto mantém um baixo impacto do desempenho. Tendo sido implementada utilizando a plataforma *Java*, o *middleware* se encontra disponível para tanto computadores pessoais quanto celulares compatíveis com a plataforma *Android*. Apesar disso, o uso do *uP* permite que aplicações construídas utilizando outras plataformas também possam se comunicar de forma transparente com recursos que adotem o *uOS*.

[5] *JavaScript Object Notation.*

Possuindo uma arquitetura flexível, o *middleware* permite a expansão de suas capacidades através de *plugins* de rede e *drivers* de recursos. O uso de *plugins* de rede permite o acesso a múltiplos tipos de tecnologias de comunicação, tendo disponíveis em sua implementação versões para acesso a redes *Bluetooth* bem como *Sockets TCP* e *UDP*. Cada *plugin* possibilita a definição de meios especializados para a localização de outros dispositivos na rede, além da forma como serão gerenciados os canais de comunicação. No caso dos *Sockets*, as estratégias de descoberta presentes incluem tanto a observação da tabela *ARP* (limitante a um nível de nós presentes em uma rede física) ou a varredura de endereços, que apresenta tempos de respostas bastante elevados (acima de 30 segundos). Por esta razão foi desenvolvido um terceiro método[6], utilizando técnicas de *multicast UDP*, reduzindo o tempo de resposta para valores abaixo de 3 segundos.

Diversas topologias podem ser aplicadas quando se deseja conectar um conjunto de nós em uma rede. Dentre elas, duas estratégias se destacam: de forma descentralizada (ou ponto a ponto[7]) onde cada nó controla a sua comunicação e descoberta dos demais; ou centralizada, na qual um nó central possui as informações sobre outros. O *middleware* permite que ambas as estratégias sejam usadas, inclusive de forma conjunta, em um ambiente inteligente. Por padrão, toda comunicação que ocorre no ambiente segue uma topologia descentralizada. Alternativamente, um ou mais nós podem operar como "listas amarelas", denominadas de *Registers*, fornecendo suas informações para nós que possuem restrições. Apesar disso, a comunicação utilizando o *middleware* é restrita aos *plugins* de disponíveis, que hoje se limitam a redes locais. Tendo em vista a grande demanda por colaboração em regiões metropolitanas (ou mesmo globais) nos jogos ubíquos, torna-se necessário o suporte a uma rede que tenha tal grau de abrangência. Com este propósito foi construído o *HTTP-plugin*[8], que permite a troca de informações entre dispositivos através de um servidor central, podendo este se localizar na nuvem. Tal *plugin* utiliza *WebSockets* iniciados pelos dispositivos clientes, permitindo a comunicação bidirecional mesmo em topologias onde o acesso externo seja limitado[9].

4.2 uImpala Game Engine

O propósito do motor *uImpala* [69] é permitir o desenvolvimento de títulos que utilizem os dispositivos no ambiente enquanto mantém uma arquitetura familiar aos desenvolvedores de jogos. O motor permite a criação facilitada de mundos virtuais, ponto forte no desenvolvimento de jogos, que se conectem com o mundo real, ponto forte da computação ubíqua. Para isto, a *uImpala* se baseia em uma arquitetura já estabelecida para o desenvolvimento deste tipo de ferramenta, juntamente com a aplicação de boas práticas do meio. Isto facilita sua curva de aprendizado, o reuso de seus componentes e integração com outras ferramentas e bibliotecas.

A estrutura interna do *uImpala* é formada por três componentes conforme representado na Figura 4.3. O suporte à construção dos elementos lógicos do jogo é provido pelo *Núcleo Lógico* do motor. Este fornece classes e objetos, independentes de plataforma,

[6] `https://github.com/UnBiquitous/uos_socket_plugin`

[7] Do inglês *peer to peer* ou *P2P*.

[8] `https://github.com/UnBiquitous/http-plugin`

[9] A manutenção dos *WebSockets* iniciados pelo dispositivo cliente permite que mesmo nos casos em que se esteja usando técnicas como *NAT* ou bloqueio de portas externas o acesso seja possível e mantido.

Figura 4.3: Arquitetura interna do motor de jogo *uImpala*.

que auxiliam na criação de comportamentos e regras. Utilizando o *middleware uOS* o *Subsistema de Entrada e Saída* controla os dispositivos disponíveis ao jogo. Por fim a *Biblioteca de Recursos* fornece uma *API* multiplataforma para construção de elementos de mídia e conteúdo.

4.2.1 Núcleo Lógico

O elemento lógico central de um jogo são as cenas que o compõem. Estas consistem em agrupamentos de objetos de jogo, que representam seus elementos, e carregam consigo as regras que formam sua mecânica. Tanto cenas como objetos de jogo possuem comportamentos próprios definidos pelos desenvolvedores. É através da colaboração entre estes objetos, entradas do usuário e o raciocínio do sistema que se determina a mecânica sendo executada. Um jogo pode ser composto por uma ou mais cenas, porém, apenas uma destas pode estar ativa em um determinado momento. São fornecidas duas operações para estabelecer o fluxo entre cenas: comutação e empilhamento. Quando duas cenas são comutadas, o estado da cena anterior é descartado e a atual é colocada como ativa. Já quando ocorre o empilhamento, a cena anterior é desativada e a atual é ativada. Neste caso, o estado da cena anterior é mantido no topo da pilha de execução de onde pode ser retomada sua execução *a posteriori*.

O motor realiza o controle das cenas ativas e dos eventos ocorridos através de um laço de execução principal de seis estágios. Cada execução deste laço é conhecida como um ciclo de atualização e tem forte impacto na percepção que o usuário tem do jogo. Por esta razão a frequência destes ciclos é determinada pelo programador e, quando possível, assegurada pelo motor de jogo. Seis estágios compõem este laço:

1. Informações das entradas são capturadas dos dispositivos de entrada habilitados, sejam eles locais ou remotos.
2. Notificações assíncronas acerca das entradas são registradas aos *listeners* cadastrados.
3. Todas as cenas são notificadas a executar suas atualizações lógicas.

4. Todas as cenas são notificadas a executar suas atualizações de saída.
5. Informações de saída são atualizadas de acordo com as solicitações realizadas.
6. Se algum recurso for marcado para ser descartado, este é notificado e removido.

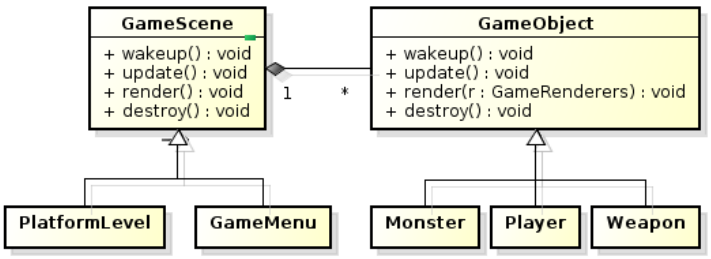


Figura 4.4: Classes de base que compõem o núcleo da *uImpala*.

As classes de base para a criação de cenas e objetos de jogo estão representadas na Figura 4.4. Através da especialização destas classes é possível definir o comportamento do jogo. Para tal, devem ser respeitadas as interfaces de cada classe, onde cada um dos métodos representa uma das notificações realizadas durante o laço de execução principal, conforme descrito a seguir:

- **Wakeup**: Método invocado quando uma cena é removida da pilha de cenas e colocada como ativa.
- **Update**: Responsável por realizar as requisições de atualização do estado através de sua lógica interna. Isto inclui cálculos de colisão, atualizações de acordo com as entradas e estímulos externos, bem como a definição das reações do jogo. Aqui é possível a criação(ou remoção) de objetos de jogo conforme necessário.
- **Render**: Define quais serão as requisições de atualização das saídas. Isto é realizado de acordo com as opções de estímulos habilitadas pelo motor de jogo, não se restringindo apenas a áudio e vídeo.
- **Destroy**: Método invocado quando uma cena é descartada da pilha de cenas. Desta forma, sendo o local onde deve-se realizar a limpeza de componentes e recursos alocados.

A Listagem 4.1 apresenta um exemplo de como um objeto de jogo pode ser criado. Nela vemos a especialização do método *update* para implementar uma lógica de ataque do personagem para apenas quando seu inimigo (*target*) esteja ao seu alcance. Já na especialização do método *render* vemos que a cada atualização, a forma (*shape*) do objeto é renderizada de acordo. O ciclo de vida deste objeto é controlado por sua cena e esta pelo ciclo de atualização central.

Assim como uma cena possui um conjunto de objetos de jogo, um objeto pode por si próprio ser composto por um conjunto de outros objetos. Isto resulta na construção de uma árvore de objetos onde cada nó não folha é, na realidade, um objeto complexo. Através da *API* disponível, os eventos disparados pelo motor durante o laço de execução principal são repassados a cada elemento desta árvore de forma a manter a lógica

```
1  public class CarnivoreSect extends GameObject {
2
3    [...]
4
5    public void update() {
6      Something target = target();
7      if (target != null && insideInfluenceRadius(target)){
8        if(sect.energy() < 5*initialEnergy){
9          sect.attack();
10       }
11     }
12     waitToMateAgain --;
13   }
14
15   public void render(GameRenderers renderers) {
16     [...]
17
18     shape.center(position());
19     shape.rotate(rotationAngle());
20     shape.render();
21
22   }
23
24   [...]
25 }
```

Listing 4.1: Exemplo de um objeto de jogo usando o motor uImpala.

de execução consistente entre todos. Objetos que compõem a árvore possuem acesso a um componente que lhes permite definir sua prioridade na atualização da saída escolhida. Dessa forma é possível determinar o resultado do estimulo ao usuário de forma independentemente da construção da árvore de objetos.

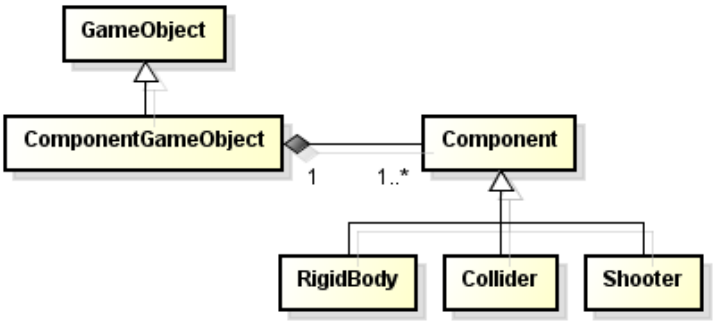


Figura 4.5: Objeto de jogo utilizando o modelo de componentes suportado pela *uImpala*.

Através da especialização das classes de objetos de jogo, é possível o reuso vertical dos comportamentos desenvolvidos. Porém, objetos que não se caracterizam por uma relação de especialização podem também compartilhar lógicas específicas. Tomando por exemplo o caso de um jogo de plataforma, tanto um bloco (obstáculo) quanto um monstro são objetos passíveis de colisão, tanto entre si como com outros objetos, porém estes não são especializações um do outro. Para facilitar o reuso nestas situações a *uImpala* suporta o desenvolvimento de objetos de jogo que seguem o modelo de composição de componentes [19] [32]. A Figura 4.5 apresenta um exemplo deste modelo em ação, onde o comportamento de um objeto é determinado pelo conjunto de componentes que o formam. A flexibilidade presente neste modelo se mostra bastante benéfica na construção de jogos que possuem um alto grau de compartilhamento de comportamento entre objetos distintos. Também é válido ressaltar que, devido aos comportamentos serem passíveis de sofrer alteração em tempo de execução, existe também a flexibilidade de se criar uma maior variabilidade de como estes objetos operam.

Jogos desenvolvidos utilizando a *uImpala* que necessitem de acesso a recursos da plataforma sob a qual estão executando, do *middleware* ou outros não estejam acessíveis pela *API* padrão, podem fazê-lo através da classe *GameSingletons*. Nesta é possível encontrar a instância corrente do *gateway* (classe de acesso a instância do *uOS*) e do jogo (uma aplicação *uOS*). Através dela é possível modificar os Gerenciadores de Entrada e Saída disponíveis em tempo de execução, conforme desejado. Além disso, esta classe pode ser utilizada como "quadro negro" para a comunicação de estados globais entre cenas e objetos distintos do jogo sendo desenvolvido.

4.2.2 Subsistema de Entrada e Saída

Motores de jogo tradicionais lidam com o tratamento de diversos tipos de entrada e saída. Mesmo que diretamente ligados ao dispositivo utilizado, se faz necessária a escolha da forma como será feita a interação (utilizando controles, gestos, *mouses* ou teclados) e o estímulo aos usuários (seja por áudio, vídeo ou retorno háptico). Nos casos em que é possível a permutação entre opções disponíveis, é comum oferecer meios para mapear as

```
1  public class Player extends GameObject {
2
3    [...]
4
5    public Player(){
6      KeyboardManager manager =
7          GameSingletons.get(KeyboardManager.class);
8      if(manager != null){
9        manager.connect(KeyboardSource.EVENT_KEY_DOWN,
10         new Observation(){
11           protected void notifyEvent(Event event,
12                         Subject subject) {
13             this.key = event.getUnicodeChar();
14           }
15         });
16       manager.connect(KeyboardSource.EVENT_KEY_UP,
17         new Observation(){
18           protected void notifyEvent(Event event,
19                         Subject subject) {
20             this.key = null;
21           }
22         });
23     }
24   }
25
26   [...]
27 }
```

Listing 4.2: Exemplo de um objeto de jogo acessando o recurso de teclado.

ações do jogo aos comandos existentes. Em um jogo ubíquo, esta tarefa se apresenta ainda mais complexa, já que muitas vezes entradas e saídas escolhidas possuem direta correlação com a metáfora sendo aplicada. Os dispositivos usados vão além daqueles disponíveis no início do jogo e conectados ao dispositivo onde a sessão está sendo executada, pois podem ser alterados durante a sessão, com o surgimento de novos dispositivos, ou com a indisponibilidade daqueles presentes.

Para controlar os recursos acessíveis no ambiente através do *uOS* são utilizados gerenciadores de entrada e saída. Estes observam as alterações de disponibilidade dos recursos, permitindo sua utilização no jogo de acordo. Eles também realizam a conversão das interações para o padrão de notificações lógicos do motor, tornando o acesso a recursos locais e remotos transparentes ao desenvolvedor. A este é possível tanto habilitar os gerenciadores desejados bem como desenvolver seus próprios, seguindo a necessidade de cada título. Na versão corrente do *uImpala* são disponibilizados gerenciadores para as entradas e saídas mais tradicionais como: *mouse*, teclado, áudio e vídeo.

A Listagem 4.2 apresenta como um objeto de jogo pode acessar o recurso de teclado e utilizá-lo durante a sua execução. Neste exemplo, qualquer teclado disponível no ambiente

será aceito como forma de entrada. Porém, utilizando o parâmetro *Subject* é possível saber qual recurso (e por consequência o dispositivo) que gerou tal entrada. Desta forma é transparente, e de escolha do desenvolvedor, qual a granularidade de acesso que deseja ter ao ambiente e a integração de seus recursos.

4.2.3 Biblioteca de Recursos

Recursos[10], dentro do motor de jogo *uImpala*, são informações de conteúdo e mídia utilizados pelo título. Exemplos incluem efeitos sonoros, imagens, vídeos e esquemas de mapas. Muitos destes recursos impõem um alta carga em termos de consumo de memória, o que torna a sua gestão de grande importância para garantir um fluxo suave de execução do jogo. Esta biblioteca é responsável por controlar não só pela criação destes recursos, mas também sua gestão, *cache* e remoção do sistema.

Recursos de baixo nível são utilizados para representar um conteúdo binário autocontido, estes são gerenciados e armazenados internamente ao motor. Apenas uma instância de cada um destes é carregada em memória, otimizando assim o consumo desta por parte do jogo. O desenvolvedor tem acesso então a recursos de alto nível, estes referenciam os recursos de baixo nível e são responsáveis pela implementação de sua conversão em estímulos de saída. Um exemplo de relação entre recursos de alto e baixo nível pode ser visto em estímulos como animações e *sprites* que se utilizam de imagens. Além disso, o estímulo de um recurso de alto nível se dá através da referência a um meio de saída, como uma tela.

Cada cena do jogo possui seu próprio contêiner de recursos. Desta forma, cada recurso é carregado em memória apenas uma vez por cena. Como parte da cena, objetos de jogo também podem carregar recursos próprios durante sua execução. Quando uma cena é descartada, todos os recursos a ela atrelados (e a seus objetos) são igualmente descartados. A biblioteca de recursos contém a implementação de um conjunto de recursos de alto nível mais familiares aos desenvolvedores de jogos. Entre estes estão inclusos: *sprites* (arquivos de imagem), animações (sequência de imagens), áudios, textos (arquivos de fontes), *titlesets* (seccionamento de imagens) e *titlemaps* (representação textual de mapas). Novos tipos de recursos, bem como especializações dos existentes, podem ser desenvolvidos e agregados ao motor conforme necessário.

4.2.4 Implementação

Cada plataforma de software provê *APIs* distintas para que seus recursos, como entrada e saída, sejam utilizados. Isto tem impacto direto em como é realizada a gestão das funcionalidades disponíbilizadas pelo motor. Por esta razão, a *uImpala* permite que entradas e saídas sejam especializadas para cada plataforma de acordo com as interfaces fornecidas. Atualmente são disponibilizadas implementações tanto para *JSE*, ambiente *desktop*, como *Dalvik*, ambiente *Android* para *smartphones* e *tablets*.

Um jogo desenvolvido utilizando apenas os componentes fornecidos pelo núcleo lógico da *uImapala* pode ser executado de forma independente, sem se preocupar com qual será a plataforma final. Para tal, no momento de compilação (ou empacotamento) deve-se especificar qual o ambiente de destino que será utilizado. Isto permite que sejam

[10]Neste caso recursos vem do inglês *assets* e não *resources*.

estabelecidas as ligações corretas para a fábrica de recursos e gerenciadores de entrada e saída relativos a cada plataforma.

O motor utiliza os padrões abertos *OpenGL* e *OpenAL* para gestão dos contextos responsáveis pelos estímulos visuais e auditivos, respectivamente. Na plataforma *Android* estes contextos são suportados pela implementação das *APIs OpenGL ES*[11] e *OpenSL ES*[12]. As bibliotecas *LWJGL*[13] e *Slick-Util*[14] são responsáveis por este suporte na plataforma *JSE*. Em ambas, as interações são capturadas e gerenciadas pelos gestores de entrada ativos. Na versão corrente são suportadas ações disparadas por *mouses*, teclados e telas de toque.

4.3 *Unity Plugin*

Dentre os motores de jogo disponíveis comercialmente o *Unity 3D*™[15] é um de uso mais difundido [16]. Suas características mais marcantes são o suporte ao desenvolvimento multiplataforma e o apoio provido por sua ferramenta de autoria. Razão esta que leva, inclusive, alguns jogos ubíquos a utilizar este motor no seu desenvolvimento [96]. O *Unity Plugin* [76] [17] provê acesso a recursos do *uOS* presentes no ambiente para jogos desenvolvidos usando este motor de jogo.

Este *plugin* opera como uma biblioteca de apoio, implementando as operações definidas pelo conjunto de protocolos *uP*. Como o *plugin* é o mesmo utilizado pelo *middleware uOS*, garante-se a operação transparente entre aplicações que utilizem um ou outro. Tendo em vista que o *middleware* oferece um conjunto bastante extenso de funcionalidades e considerando limitações existentes no ambiente do motor *Unity 3D*, as funcionalidades do *plugin* se restringem a um subconjunto do uOS:

- Conjunto completo entidades e operações do protocolo *uP*.
- Capacidade de comunicação utilizando *Sockets TCP*.
- Descoberta e gestão de dispositivos.
- Descoberta e gestão ponto a ponto (*P2P*) de recursos.
- Comunicação síncrona e assíncrona.
- Integração entre aplicações

Esta funcionalidades permitem que um jogo observe o ambiente ao seu redor e identifique quais dispositivos e recursos estão disponíveis. Estas também possibilitam que um artefato computacional que utilize-se do *plugin* publique e disponibilize seus próprios recursos e serviços a outras aplicações existentes. Isto concede a jogos desenvolvidos que

[11] http://www.khronos.org/opengles/
[12] http://www.khronos.org/opensles/
[13] www.lwjgl.org
[14] http://slick.ninjacave.com/slick-util/
[15] `http://unity3d.com/`
[16] De acordo com a própria empresa responsável por seu desenvolvimento, o motor detém 45% do mercado mundial de motores de jogos. `http://unity3d.com/public-relations`
[17] `https://github.com/lhsantos/unity_uos_plugin`

utilizem este suporte serem parte integrante de um ambiente que se adapta dinamicamente aos seus elementos.

A implementação do *plugin*, diferentemente do *middleware*, foi realizada utilizando a plataforma C# /.NET5[18] suportada pelo motor. Apesar desta ter suporte a um modelo de linhas de execução (*threads*) similar ao oferecido por outras máquinas virtuais, a *Unity 3D* restringe seu uso. Por esta razão, os componentes implementados pelo *plugin* são executados como parte do laço de execução principal através de rotinas de *callback*. Sendo assim, toda comunicação realizada é gerenciada por uma fila mensagens processada ao final de cada iteração do laço.

4.4 Execution Driver

O suporte a criação de modificações, também conhecidas como *MODs*, está presente em diversos jogos, permitindo que desenvolvedores alterem seu comportamento. Jogos como *Half-Life*[19] e *Warcraft*[20] são exemplos conhecidos de como uma grande variedade de *MODs*, além de uma forte comunidade, pode ser benéfica à longevidade de um título. Essa possibilidade de modificar o jogo coloca o usuários como elementos de mudança na experiência do jogar, uma liberdade que favorece características de *design* emergente [38].

Apesar destes jogos (MODs) permitirem que seus comportamentos sejam modificados, essa capacidade não é parte integral da mecânica de cada um. De fato, grande parte das modificações são consideradas como títulos distintos daqueles aos quais estão baseados. Por outro lado, jogos como *RoboCode*[21] requer que seus jogadores codifiquem o comportamento de seus personagens (os robôs). Tal código atua como a interface de interação e controle do usuário sobre o resultado da arena de batalha, onde competem os robôs de todos os participantes. Neste caso porém, o código é inserido antes de cada partida, não permitindo sua alteração durante a sessão. Esta limitação não ocorre nos títulos *Code Combat* [22] e *Light-Bot* [23] , nos quais o código de entrada é ajustado durante a execução do jogo.

A habilidade de modificar o comportamento de um software em tempo de execução é denominada por "Mobilidade de Código"[24] sendo de central importância para a criação de jogos abertos (Seção 3.2.2). Sua idéia revolve em torno do conceito de "Unidades de Execução", ou apenas *EUs*[24]. Essas unidades são entidades capazes de transportar conhecimento de execução entre nós distintos na rede. Uma das principais características de uma *EU* é seu conhecimento do nó onde está sendo executado. Isto permite adaptar sua lógica interna de acordo com a informação recebida. Essa capacidade é de grande interesse em aplicações da *ubicomp*, onde dados e lógica podem se ajustar e mover para dispositivos que melhor lhe sirvam enquanto seus usuários transitam entre diferentes ambientes [3].

EUs podem se mover no ambiente de acordo com três estratégias distintas: execução remota, código sob demanda e agentes móveis. A execução remota consiste em solicitar

[18] `http://www.microsoft.com/net`
[19] `http://orange.half-life2.com/`
[20] `http://us.blizzard.com/en-us/games/war3/`
[21] `http://robocode.sourceforge.net/`
[22] `http://codecombat.com/`
[23] `http://light-bot.com/`
[24] Do inglês *Execution Unities*

que uma *EU* realize uma operação em um nó distinto, comumente retornando resultados ao nó de origem. Tal estratégia pode ser aplicada no intuito de poupar a utilização da rede, transferindo uma tarefa de intenso processamento para um local mais próximo dos recursos por ela utilizados. Já o código sob demanda opera de forma oposta, onde o nó requisitante solicita a transferência de uma *EU*. Isto permite que este nó adquira capacidades não presentes anteriormente, como a habilidade de converter um dado de formato desconhecido em um que seja capaz de interpretar. Agentes móveis consistem na estratégia mais flexível, possibilitando que *EUs* trafeguem entre nós conforme desejam, levando consigo seus dados além de sua lógica. Estas três modalidades estão acessíveis através de dois serviços providos pelo *driver* (*"remoteExecution"* e *"executeAgent"*) em conjunto como uma entidade para a representação de *EUs* (*"ExecutionUnity"*) conforme descrito a seguir.

4.4.1 *Execution Unities*

Utilizando a classe *ExecutionUnity* desenvolvedores podem criar unidades de execução passíveis de serem transportadas entre nós da rede. Estas unidades carregam consigo tanto o código que determina seu comportamento como os dados que compõem seu estado. Para a definição da lógica de execução são utilizados *Scripts Lua* [52]. Isto permite que a unidade seja transportada dentro de mensagens *uP*, juntamente com seu estado. A Listagem 4.3 apresenta como uma *ExecutionUnity* pode ser construída. Nesta, é possível observar que a criação de atributos de estado é transparente ao desenvolvedor, como no caso da variável *state*. Adicionalmente funções auxiliares podem ser desenvolvidas e habilitadas ao desenvolvedor, expandindo assim as capacidades acessíveis as unidades. A flexibilidade dos objetos desta classe permite que sejam utilizados tanto na estratégia de código sob demanda como para a construção de agentes móveis.

```
1 StringBuffer script = new StringBuffer();
2 script.append("state = 0 \n");
3 script.append("function addTwo() \n");
4 script.append("     state = state + 2 \n");
5 script.append("     return state \n");
6 script.append("end\n");
7 ExecutionUnity ex = new ExecutionUnity(script.toString());
8 System.out.println("Value: "+ex.call("addTwo"));
```

Listing 4.3: Exemplo de como uma *Execution Unity* pode ser criada.

4.4.2 *Execute Agent*

Este serviço do *driver* permite a transferência de agentes, juntamente com seu estado, descritos através da linguagem *Java*. Para isso é utilizado o suporte fornecido pela *JVM* para a serialização de objetos em conjunto com a ferramenta *ClassToolbox*, desenvolvida como parte do *driver*. Essa ferramenta permite que classes das quais um agente possua dependência sejam descobertas em tempo de execução e, caso não presentes no dispositivo de destino, sejam transferidas juntamente com o agente. Ela também é responsável pela

criação de *ClassLoaders* específicos para cada agente, evitando assim que suas classes interfiram com a execução das demais.

```
1 public class ExampleAgent extends Agent{
2   public void run(Gateway gateway){
3     // Lógica do agente
4   }
5 }
6 . . .
7 AgentUtil.move(new ExampleAgent(),targetDevice,gateway);
```

Listing 4.4: Exemplo como um Agente Móvel e na sequencia como este pode ser transferido (linha 7).

A transferência de classes entre duas *JVMs* que interpretam o mesmo padrão de *bytecode* ocorre de forma simples e direta, bastando a transferência de seus arquivos "compilados". Isso permite que o esquema descrito para este serviço opere de forma satisfatória para casos em que ambos os nós envolvidos na comunicação estejam utilizando uma máquina virtual compatível com o *bytecode JSE*. Para permitir que essa comunicação também seja possível entre máquinas virtuais distintas, a *ClassToolbox* fornece um mecanismo para a conversão entre *bytecodes* distintos. Na atual versão é possível transformar uma classe de origem *JSE* para *Dalvik* usando a ferramenta *DEX*[25] provida pela plataforma *Android*. Isso permite que um agente seja transferido de um ambiente *desktop* para um celular de forma transparente ao programador. Apesar disso, a tal agente não é permitido fazer uso, ou referência, a classes específicas a *API* da plataforma, já que estas não estarão disponíveis no seu destino.

4.4.3 *Remote Execution*

A execução remota, de maneira similar as unidades de execução, se utiliza de *scripts Lua* para transferir lógica computacional entre nós distintos. A Listagem 4.5 demonstra como este serviço pode ser utilizado para se delegar a execução de um algoritmo para um outro dispositivo. Ao final da execução deste, os dados computados são retornados ao nó requisitante como parte da resposta do serviço. Para a interpretação dos *scripts Lua* foi utilizada a implementação provida pela biblioteca *LuaJ*[26] portável tanto para *JSE* quanto *Dalvik*.

```
1 StringBuffer script = new StringBuffer();
2 script.append("a = Uos.get(UOS_ID,'a') \n");
3 script.append("b = Uos.get(UOS_ID,'b') \n");
4 script.append("Uos.set(UOS_ID,'sum',a+b) \n");
5
6 Call call = new Call("uos.ExecutionDriver","remoteExecution");
7 call.addParameter("a",2);
8 call.addParameter("b",3);
9 Response r = gateway.callService(call);
```

[25] https://source.android.com/devices/tech/dalvik/dex-format.html
[26] http://luaj.sourceforge.net/

```
10 System.out.println("Value: "+r.getResponseParam("sum"));
```

Listing 4.5: Exemplo de como uma execução remota pode ser solicitada.

4.5 Mudanças no uOS

Além da construção de componentes externos complementares ao *middleware*, como o motor de jogo *uImpala* e o plugin de rede *HTTP*, mudanças internas a arquitetura do middleware foram realizadas durante a pesquisa visando três objetivos: maior compatibilidade entre plataformas[27], maior flexibilidade no acesso a serviços e otimização no uso de rede. Estas modificações podem ser encontradas na versão disponível do *middleware* de número 3.1 .

O *middleware* já possuía uma arquitetura modular na qual tanto a camada de rede, os *drivers* de serviço e aplicações compunham partes externas ao núcleo, sendo estes apenas suportados por ele. Apesar disto, alguns de seus componentes ainda possuíam uma dependência de bibliotecas externas. O caso de maior destaque se encontra nos componentes de gestão de *drivers*, gestão de dispositivos e autenticação básica. Esses se utilizavam[28] da biblioteca de persistência *HSQLDB*[29]. Essa biblioteca fornecia aos gestores mecanismos de armazenamento em memória dos dados de recursos e dispositivos presentes no ambiente. A não persistência física se justifica já que estes dados são efêmeros e dependem do estado corrente do *smart space*. A retenção desta informação foi substituída por um novo componente, fazendo uso de "mapas *hash*" para o acesso aos dados. Já no caso do *plugin* de autenticação a persistência dos dados é necessária, já que as chaves de acesso devem ser mantidas entre execuções distintas, motivo pelo qual este componente foi externalizado ao núcleo do *middleware*.

Buscando maior flexibilidade no acesso aos serviços, duas novas funcionalidades foram acrescidas ao núcleo. A primeira consiste na relação de equivalência entre *drivers*, conforme descrito anteriormente. A implementação ocorreu no componente gestor de *drivers* através da inserção de múltiplas entradas de acordo com a árvore de equivalência informada. Isso é feito apenas de uma vez, tendo em vista que as interfaces de recursos não permitem alteração em tempo de execução. O segundo recurso consiste na possibilidade de invocação e descoberta de serviços relacionados a aplicações. Tais serviços operam de forma similar aos providos por recursos, porém são mais adequados para capacidades não compartilhadas além da aplicação em questão.

A otimização do uso da rede se faz necessária já que este é o meio por onde toda comunicação ocorre no ambiente. Conforme exposto anteriormente, a adição ao *Socket Plugin* do *Multicast Radar* visou uma melhoria nos tempos de descoberta de dispositivos. Além disso, outra alteração foi a criação de um componente que permite a prevalência das conexões, conforme a necessidade de cada *plugin* de rede. Este componente é denominado de *Cache Controller* e foi originalmente construído para ser utilizado no *Bluetooth Plugin*

[27] No caso, o maior foco foi possibilitar que o mesmo núcleo do *middleware* fosse passível de ser utilizado no ambiente da máquina virtual *Dalvik* presente na plataforma *Android* sem a necessidade de personalizações.

[28] Esta dependência é incompatível com o ambiente provido pelo *Android* devido ao conflito com os recursos já providos pela biblioteca de persistência *SQLite*. `www.sqlite.org`

[29] `http://hsqldb.org/`

devido ao alto custo de estabelecimento de conexões neste meio. O componente permite que conexões sejam reutilizadas em múltiplas trocas de mensagem, evitando o custo de uma nova abertura. Para isso, toda vez que uma conexão de controle tem seu término solicitado, a ela é dado um tempo de reaproveitamento[30] no qual a conexão é mantida. Conexões de dados não são reaproveitadas tendo em vista que seu uso não pode ser compartilhado entre chamadas distintas.

4.6 Hierarquia de Recursos

O ambiente inteligente é dinâmico não só em elementos que nele aparecem mas também com relação a evolução destes. Novos tipos de dispositivos e interações são criados continuamente, sendo de interesse do ambiente inteligente que estes também sejam embarcados como parte de suas capacidades. No caso dos jogos ubíquos, conhecer quais são estes tipos de artefatos e o que eles têm a oferecer é de suma importância em seu projeto e desenvolvimento. Para *game designers*, conhecer quais são as possibilidades de interação e seus agrupamentos já é de grande valia na elaboração de melhores metáforas que liguem o mundo real ao virtual. Já para desenvolvedores, é necessário conhecer quais interfaces essas formas de interação possuem. Isso favorece não só na padronização do acesso a tais recursos como também na construção de *softwares* de acesso a novos tipos de forma reusável.

No caso dos jogos abertos, o conhecimento destas interfaces contribui para que seja possível permutar entre recursos equivalentes entre si. Por esta razão, e com base nas listas de capacidades exposta no Capítulo 3, foi elaborada a *hierarquia de recursos* representada na Figura 4.6. Na figura estão apresentadas interfaces de acesso a diversos tipos de entradas e saídas de uso comum entre jogos ubíquos. A definição dos serviços estabelecidos em cada uma destas interfaces é melhor detalhada a seguir:

- **Usuário**: Fornece serviços contendo a identificação dos usuários do ambiente.
  - *Lista de Presença*: Fornece a lista de usuários presentes no ambiente no momento da consulta.
  - *Usuário Entrou*: Informa que um novo usuário foi detectado no ambiente.
  - *Usuário Partiu*: Informa que um usuário deixou de ser observado no local.
- **Objeto**: Fornece serviços informando a identificação dos objetos reconhecidos no local.
  - *Lista de Presença*: Fornece a lista de objetos encontrados no ambiente no momento da consulta.
  - *Objeto Entrou*: Informa que um objeto foi reconhecido no ambiente.
  - *Objeto Partiu*: Informa que um objeto não é mais detectado no local.

  Interação com o Objeto: Expande os serviços incluindo informações acerca da interação com o objeto em questão.

[30] Atualmente este tempo consiste em três minutos, sendo passível de configuração pelo programador

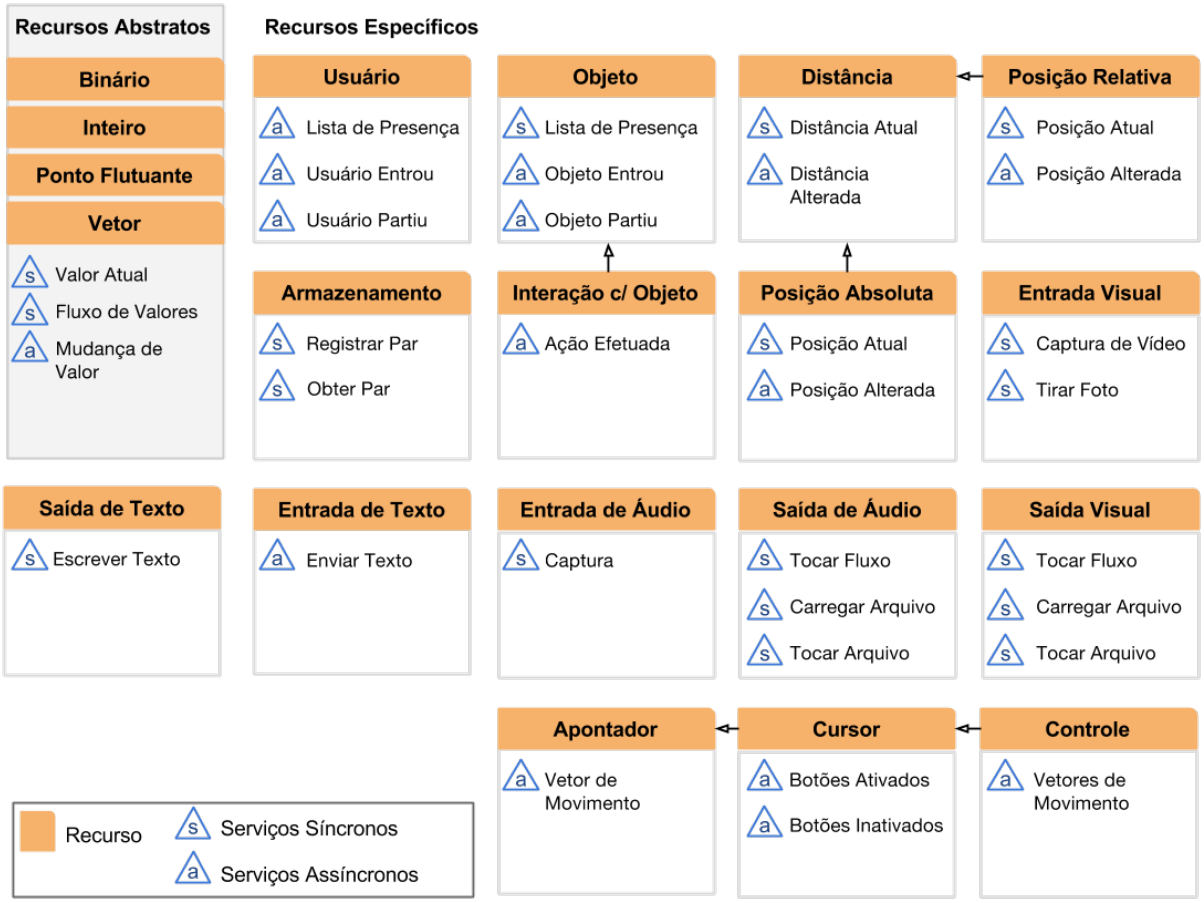


Figura 4.6: Hierarquia de recursos para jogos ubíquos.

  - *Ação Efetuada*: Informa que determinada ação foi executada no objeto desejado.

- **Distância**: Fornece serviços que provam a informação de distância entre dois elementos.

  - *Distância Atual*: Fornece a distância atual entre dois elementos observados.
  - *Distância Alterada*: Informa que a distância entre dois elementos se modificou.

  Posição Relativa: Fornece serviços que informam a posição, em três dimensões, relativa a um ponto conhecido.

  - *Posição Atual*: Fornece a posição atual do elemento observado.
  - *Posição Alterada*: Informa que a posição do elemento foi alterada.

  Posição Absoluta: Fornece serviços que informa a posição, em três dimensões, em coordenadas no globo terrestre.

  - *Posição Atual*: Fornece a posição atual do elemento observado.
  - *Posição Alterada*: Informa que a posição do elemento foi alterada.

- **Armazenamento**: Fornece serviços que permitem persistir informações no formato chave/valor no dispositivo.
  - *Registrar Par*: Permite que um par contendo uma chave e um valor seja persistido.
  - *Obter Par*: Fornece o valor atrelado à chave informada.
- **Saída de Texto**: Fornece serviços para acesso a saídas de texto.
  - *Escrever Texto*: Recebe um conjunto de caracteres e os reproduz no dispositivo.
- **Entrada de Texto**: Fornece serviços de captura de entradas textuais.
  - *Enviar Texto*: Informa o texto detectado, ou informado, pelo usuário.
- **Entrada de Áudio**: Fornece serviços de captura de ondas sonoras.
  - *Captura*: Realiza a captura e transmissão dos dados sonoros capturados pelo dispositivo no formato informado.
- **Saída de Áudio**: Fornece serviços de estímulos sonoros.
  - *Tocar Fluxo*: Recebe um fluxo de áudio e o reproduz no dispositivo.
  - *Carregar Arquivo*: Recebe um arquivo de áudio e carrega na memória do dispositivo.
  - *Tocar Arquivo*: Reproduz um arquivo de áudio em memória.
- **Entrada Visual**: Fornece serviços de captura de imagens.
  - *Captura de Vídeo*: Realiza a captura de um fluxo de imagem no dispositivo.
  - *Tirar Foto*: Captura uma única imagem no momento da chamada.
- **Saída Visual**: Fornece serviços de estímulos visuais.
  - *Tocar Fluxo*: Reproduz um fluxo de vídeo no dispositivo.
  - *Carregar Arquivo*: Recebe um arquivo de vídeo e carrega na memória do dispositivo.
  - *Tocar Arquivo*: Reproduz um arquivo de vídeo em memória.
- **Apontador**: Fornece serviços relacionados à captura de movimentos bidimensionais.
  - *Vetor de Movimento*: Fornece um vetor contendo as distâncias propagadas em dois eixos durante uma interação.

  Cursor: Estende os serviços incorporando ações ativadas por botões.
  - *Botões Ativados*: Informa quais botões foram pressionados em um momento.
  - *Botões Inativados*: Informa quais botões deixaram de ser pressionados em um momento.

Controle: Estende os serviços incorporando múltiplos vetores de interação.

- *Vetores de Movimento*: Fornece uma lista de vetores contendo as distâncias propagadas durante uma interação.

Além destes, são definidos quatro outros recursos complementares denominados "Recursos Abstratos". Estes estabelecem as interfaces para o acesso a tipos de dados básicos. Estes dados não requerem correlação semântica junto a origem da informação, sendo bastante úteis na aplicação em "jogos artísticos"[31]. Estas aplicações podem coletar dados vindos de diferentes fontes e ajustar seu propósito de acordo com a intenção do artista. Por exemplo, um fluxo de inteiros pode ser usado para se criar a sequência de obstáculos que um jogador deve superar. Tais valores podem ser obtidos tanto de um sensor de temperatura como da luminosidade do ambiente, sem necessitar o ajuste da aplicação. Os tipos de dados suportados são: Binário, Inteiro, Ponto Flutuante e Vetores. Cada um destes está acessível através de três serviços:

- *Valor Atual*: Fornece o valor corrente no dispositivo.
- *Fluxo de Valores*: Fornece um fluxo com os valores sendo observados no dispositivo.
- *Mudança de Valor*: Informa quando o valor foi alterado.

Utilizando estes recursos é possível se desenvolver *drivers* que implementem essas interfaces de maneira que possam ser utilizadas em diversos jogos de forma transparente. Tal capacidade é de grande importância no caso de jogos abertos, nos quais dois recursos de mesmo tipo podem ser permutados sem se alterar o jogo. Além disso, novos recursos podem ser criados, expandindo os serviços prestados. Mesmo assim, se estes respeitarem a hierarquia aqui definida, poderão ser utilizados por aplicações já existentes sem a necessidade que qualquer alteração seja realizada.

[31] Do inglês *gameart*.

Capítulo 5

Avaliação

Os jogos ubíquos reconfiguráveis têm como propósito expor novas formas de utilização dos dispositivos do ambiente, enquanto a plataforma *uOS* se empenha em fornecer-lhes suporte. Neste capítulo são descritas as etapas de avaliação intermediárias e finais de ambos. Concomitante à construção da plataforma uOS (e do motor de jogo *uImpala*) foram desenvolvidos por pesquisadores do grupo de pesquisa UnBiquitous dois jogos que serviram para identificação e validação das funcionalidades da plataforma. Esta etapa é descrita na Seção 5.1. Em um segundo momento foram convidados doze desenvolvedores para projetar e construir jogos utilizando os conceitos de jogos ubíquos e, em especial, reconfiguráveis. Este grupo projetou nove jogos, dos quais quatro chegaram a ser implementados. Considerando a relevância dos nove projetos para esta tese, na (Seção 5.2) são apresentados os cinco jogos que ficaram restritos à fase de conceituação, e na (Seção 5.3) os quatro jogos que chegaram a ser prototipados.

Complementar a estes experimentos também é apresentado o jogo *uSect* (Seção 5.4), que exemplifica a construção de um "Jogo Aberto" onde tanto os dispositivos de entrada como seus comportamentos podem ser adaptados por seus usuários.

Por fim a Seção 5.5 apresenta os resultados da avaliação quantitativa realizada acerca do funcionamento da plataforma em um ambiente real.

5.1 Elaboração da *uImpala*

Nesta etapa inicial, foram criados os primeiros experimentos de aplicação prática dos conceitos de reconfiguração. Em especial, se avaliou como dispositivos podem ser incorporados na concepção de jogos ubíquos reconfiguráveis. Através da implementação deste tipo específico de jogo foi possível identificar quais funcionalidades o *middleware uOS* deveria ter para ser efetivamente um agente de suporte na construção de jogos. Além disso, essa fase identificou quais características seriam necessárias a um motor de jogos que o complementasse. Como resultado desta etapa obteve-se, além dos objetivos elencados, dois jogos construídos, o "*Run Fast!*" e o *uMoleHunt*.

O "*Run Fast!*"[1] [29] foi o primeiro jogo desenvolvido e consiste em um simulador de corrida de carros que é visualizado em uma tela pública (Figura 5.1), como um monitor ou televisão, onde é exibida uma pista de corrida em visão aérea. Dois times competem

[1] `https://github.com/rafasimao/RunFast`

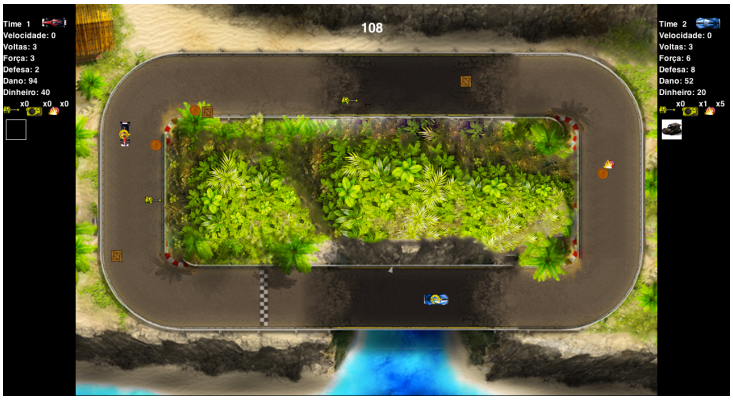
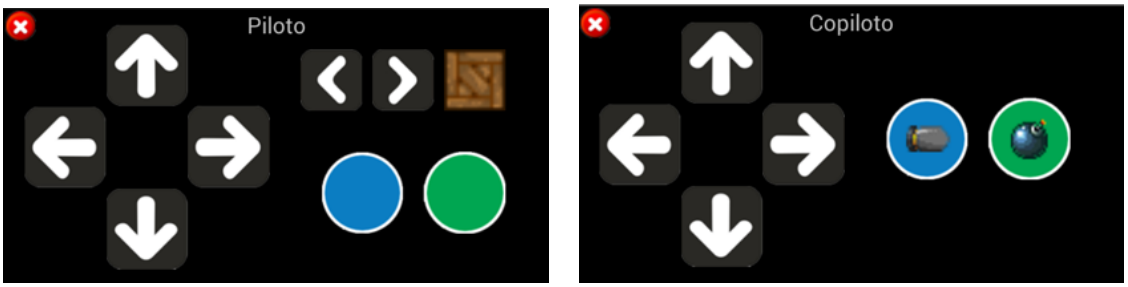


Figura 5.1: Tela pública do jogo *Run Fast!*.[29]

por quem finaliza o percurso no menor tempo, sendo que a interação com o jogo se dá através das telas dos celulares dos jogadores presentes no ambiente.



(a) Controle do piloto.

(b) Controle do co-piloto.

Figura 5.2: Dois tipos de controles presentes no jogo "*Run Fast!*". [29]

Apenas um jogador assume o papel de piloto, restando ao segundo membro da equipe o papel de co-piloto e a um possível terceiro, a função de mecânico. Cabe ao piloto a tarefa de manobrar o carro através do percurso apresentado na tela. Suas preocupações incluem também a coleta de itens que surgem na pista, bem como desviar de obstáculos e armadilhas. O co-piloto fornece suporte durante a corrida, fazendo uso dos itens coletados (Figura 5.2) a fim de auxiliar seu colega ou atrapalhar o oponente.

Por fim, fica atribuído ao mecânico a tarefa de solucionar minijogos que podem ser iniciados durante a partida. Esses consistem em eventos habilitados por meio dos itens coletados pelos pilotos e acionados pelos co-pilotos. Um dos minijogos ("*Bonus*" visto na Figura 5.3(a)) dá a chance de auxiliar o piloto de sua equipe. Já o outro ("*Break*" visto na Figura 5.3(b)) busca atrapalhar o piloto da equipe adversária.

Apesar de existir a limitação de apenas três membros por equipe e da estrita definição de papéis entre eles, o jogo "*Run Fast!*" é bastante flexível, permitindo que novos jogadores entrem durante a partida. Estes necessitam apenas possuir o aplicativo que implementa o recurso de controle em seu celular.

No "*Run Fast!*" todos os jogadores utilizam o mesmo tipo de dispositivo (seus celulares), porém cada um desempenha um papel diferente através de mecânicas distintas. Todos se integram em uma experiência conjunta através da tela pública. Além disso,

(a) Mini jogo *Bonus*. (b) Mini jogo *Break*.

Figura 5.3: Dois mini jogos presentes em "*Run Fast!*".

novos jogadores podem ser incorporados em tempo de execução, ajustando seus papéis. Neste jogo as características de integração espontânea e uso de serviços do *middleware* facilitaram o desenvolvimento e experimentação do conceito de jogos ubíquos. Contudo, seus componentes de jogo foram desenvolvidos utilizando integralmente as plataformas Android (para o celular) e JDK (para a tela pública).

Utilizando o aprendizado acumulado durante a construção do "*Run Fast!*", foi desenvolvido o título *uMoleHunt*[2] servindo de implementação guia para o motor de jogo *uImpala*. Esse jogo também faz uso de uma tela pública acessível a todos os participantes (Figura 5.4). Seus jogadores são divididos em dois times: policiais e mafiosos. O objetivo dos policiais é descobrir quem são os informantes (*moles*[3]) e obter as pistas necessárias para trazer a organização criminosa à justiça, ao passo que os mafiosos querem encontrar os informantes para evitar que seus dias de crime cheguem ao fim. Para isso, os jogadores trocam turnos utilizando os teclados de seus celulares na tentativa de descobrir os nomes uma letra por vez. Novos jogadores podem ser inseridos ao jogo a qualquer momento, sendo eles distribuídos de forma igualitária entre os times dinamicamente. O jogo termina quando um dos times consegue descobrir a identidade de dez informantes.

Apesar de ser possível incorporar novos jogadores durante a sessão de jogo, o *uMoleHunt* não modifica elementos de sua mecânica ou mesmo estética. Porém o desenvolvimento dos seus elementos visuais e de interação permitiram compor o conjunto inicial de funcionalidades providas pelo motor de jogo.

[2] `https://github.com/matheuscscp/uMoleHunt`

[3] Do inglês informal onde *mole* (toupeira) é equivalente a versões em português como "dedo-duro" ou "X9".

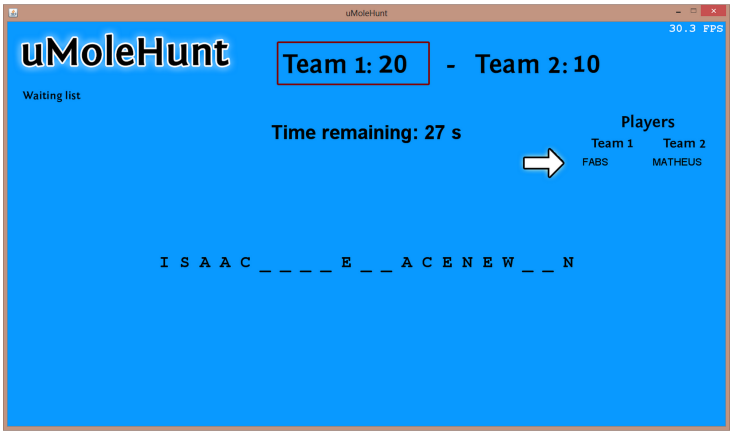


Figura 5.4: Tela pública do jogo *uMoleHunt*.

5.2 Jogos Projetados

Conforme mencionado no início deste capítulo, com o objetivo de avaliar a utilização dos conceitos de *ubigames* (e em especial os reconfiguráveis), foram convidados doze alunos participantes uma disciplina especial sobre jogos ubíquos[4]. Por um período de um mês foi apresentado ao grupo conceitos relativos ao desenvolvimento de jogos, jogos ubíquos e jogos ubíquos reconfiguráveis. Em seguida, os alunos se auto-organizaram em grupos responsáveis por elaborar títulos de criação própria, utilizando por base o conteúdo ensinado.

Ao final desta fase chegou-se ao projeto de nove títulos distintos, sendo que destes cinco se restringiram apenas ao projeto e implementação de provas de conceito, e podem ser encontrados a seguir. Os quatro títulos que foram integralmente implementados são descritos na Seção 5.3.

5.2.1 Zombie Witch of 71

Esse jogo[5] coloca seus usuários no papel dos personagens do seriado "Chaves del Ocho" enfrentando a "Bruxa do 71" quando esta se tornou um zumbi, ameaçando a vida dos habitantes da vila. Consiste em um jogo de plataforma onde cada tela pública do ambiente representa um local distinto da vila onde os personagens moram. Os usuários controlam os movimentos de seus avatares utilizando seus celulares. Cada um deles deve encontrar os itens presentes em cada um dos ambientes disponíveis para assim poderem enfrentar a "Bruxa". Caso um jogador sofra muito dano, ficará paralisado, sendo necessário que um de seus companheiros vá até onde ele esteja (virtual e fisicamente) e o libere da paralisia (Figura 5.5). Para isso o amigo deve realizar um gesto com seu celular representando o ato de jogar um balde com água.

Apesar do jogo ocorrer no mundo virtual da vila, elementos do ambiente são adaptados ao universo do jogo. Dessa forma, cada computador presente se torna um dos ambientes da vila, de modo que, apesar da mobilidade possuída pelos jogadores por usarem seus

[4]Entre os doze alunos cinco eram de graduação, seis de mestrado e um de doutorado. Dentre todos, quatro já possuíam conhecimento de desenvolvimento de jogos.

[5]`https://github.com/carlosbpf/ubigameszen`

Figura 5.5: Dois jogadores se encontram no jogo *Zombie Witch of 71*.

celulares, estes permanecem em frente à tela onde seus personagens se encontram. Essa mobilidade também é explorada ao favorecer que os jogadores se encontrem no mundo virtual, forçando os jogadores a se moverem também no mundo real.

A integração entre vários dispositivos com propósitos distintos em tempo de execução é uma das características almejadas pelos jogos reconfiguráveis. No caso do "*Zombie Witch of 71*" isso é feito de forma que os dispositivos estáticos (monitores), representam elementos fixos do jogo (as partes da vila) forçando os jogadores a entender o ambiente físico para saber como o mundo virtual está representado.

5.2.2 UbiDefense

UbiDefense[6] é um jogo do tipo "*Tower Defense*", onde jogadores devem posicionar torres de defesa em um mapa para se protegerem de monstros que os atacam. Toda a interação ocorre através do uso do celular dos jogadores. O mapa é definido como um perímetro geolocalizado, criado pelo próprio jogador ou por outros colegas. Durante a sessão, monstros tentam cruzar o mapa através do caminho mais curto entre o ponto de partida e de fim do perímetro. Para evitar isso, os jogadores devem construir torres que tanto atraem como atacam os monstros. Ao chegarem próximos a torre, os monstros a golpeiam causando dano que pode vir a destruí-la (Figure 5.6).

Como o mapa é geolocalizado, as torres são construídas utilizando a posição determinada pelo *GPS* do celular do jogador. Desta forma, obstáculos presentes no terreno escolhido se tornam desafios a serem superados pela estratégia aplicada. Além disso, a densidade e potência de sinais *WiFi* determina o alcance de cada torre. Torres dentro de zonas de rede sem fio possuem um raio de alcance maior determinado proporcionalmente à soma da força de seus sinais.

Apesar de utilizar apenas os celulares como dispositivos de interação, o *UbiDefense* explora as informações de pontos de acesso sem fio para criar uma conexão entre o jogo e o ambiente a sua volta. Isso torna cada local onde se realiza o jogo um mapa distinto já que a configuração destas redes é distinta de local para local. Essa variabilidade incentiva o jogador a experimentar novos locais em busca de experiências distintas, diferenciando o jogo de outros *LBGs*.

[6] `https://github.com/JorgeAndd/ubiDefense`

Figura 5.6: Torre sob ataque no jogo *UbiDefense*.

5.2.3 UbiÁreasDeRiscos

Este jogo[7] combina duas mecânicas contrastantes, uma competitiva e outra colaborativa, que convidam os jogadores a rever a forma como interagem com o lugar onde vivem. Ambos os modos utilizam informações de *GPS* providas pelos celulares dos jogadores[8]. No modo *Áreas* (Figura 5.7(a)), o jogador estabelece áreas de permanência no mapa de sua cidade. Quanto mais tempo o jogador ficar presente em um local, maior o círculo por ele formado. O objetivo de um jogador é ter a maior área de intersecção com os demais. Para tanto, ele deve colaborar com o maior número de jogadores diferentes para que suas intersecções prevaleçam sobre as demais.

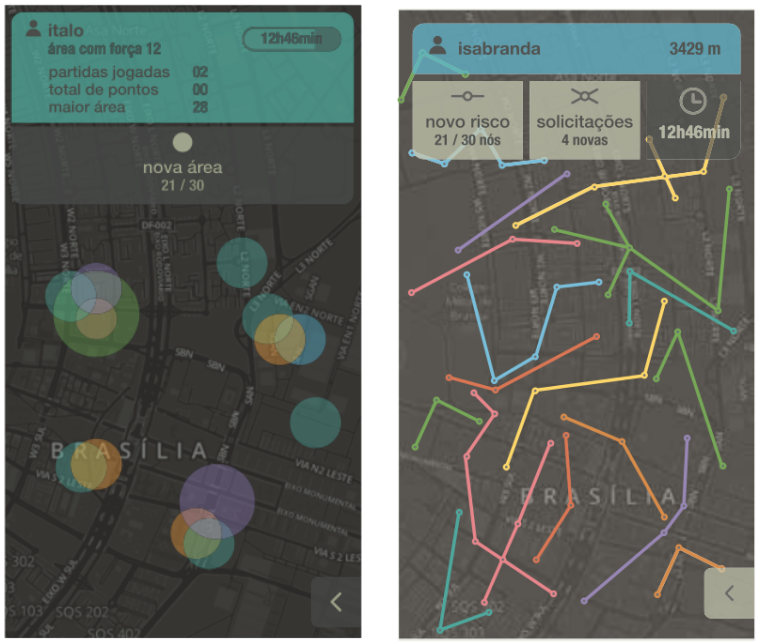
Já o modo *Riscos* (Figura 5.7(b)) funciona de maneira competitiva, sendo o objetivo do jogador possuir o maior segmento contínuo de retas. Quando duas retas pertencentes a jogadores distintos se cruzam, uma delas se rompe dando espaço a outra. As chances de uma reta ser rompida é proporcional ao seu tamanho. Desta forma os jogadores devem equilibrar retas longas (que dão maior vantagem na pontuação do jogo) e retas curtas (que possuem maior resistência quanto a serem rompidas).

O *UbiÁreasDeRiscos* é um exemplo de *LBG* tradicional, onde o elemento de interação utilizado é a localização do usuário. O diferencial aqui é convidar o jogador a repensar seus caminhos do dia a dia em função do jogo e suas estratégias.

5.2.4 DetetUbi

O jogo *DetetUbi*[9] convida os jogadores a resolverem mistérios em um mundo cercado de enigmas e segredos. Nele os jogadores podem tanto criar desafios como solucioná-los utilizando diversos elementos do ambiente. Jogadores criadores de mistérios devem definir um conjunto de tarefas que devem ser satisfeitas através dos recursos presentes nos celulares. Um exemplo de tarefa é decodificar uma mensagem secreta, o que requerirá a coleta das letras que compõem o *SSID*[10] de redes *WiFi*. Jogadores-detetives obtêm seus

[7] `https://github.com/filipepontelima/UbiAreasdeRiscos`
[8] Este jogo consiste na adaptação do *design* elaborado por Brandalise [17]
[9] `https://github.com/dedebf/DetetUbi`
[10] Do inglês "Service Set Identifier", corresponde ao nome identificador de uma rede sem fio.



(a) Modo Áreas. (b) Modo Riscos.

Figura 5.7: Dois modos presentes em "*UbiÁreasDeRiscos*".

mistérios através de "estações de desafio" presentes nos computadores dos usuários que as criaram. Somente após a resolução de um mistério prévio que um novo poderá ser então adquirido.

Apesar de utilizar os celulares como forma de interação, o *DeteUbi* é um exemplo de jogo ubíquo reconfigurável espontâneo. Em cada ambiente o jogador encontra dispositivos e informações que o auxiliam a satisfazer os mistérios a serem resolvidos. Computadores operam como bases, contendo registros de missões a serem resolvidas. Pontos de acesso sem fio representam pequenas pistas para que o jogador possa progredir na sua tarefa. Como estes dispositivos variam de local para local o jogador deve experimentar diferentes locais de forma a resolver seus mistérios.

5.2.5 UbiSoldiers

Este jogo[11] coloca seus usuários na posição de generais comandantes de tropas em uma batalha pela dominação do mundo. Utilizando apenas seus celulares os jogadores devem gerenciar três elementos: ouro, unidades, itens. Quantidades de ouro são gastas pelos jogadores sempre que estes realizam buscas nos arredores. É por meio das buscas que é possível obter novos itens e unidades para formar os exércitos. As unidades consistem nos tipos de soldados que podem ser utilizados pelo general em batalha e os itens servem para modificar os atributos das unidades, concedendo a elas vantagens estratégicas.

Ao realizar uma busca, as chances do jogador obter um item são proporcionais ao seu nível de raridade. As chances sofrem a ação de modificadores, plugáveis ao jogo dependendo dos sensores disponíveis no celular, e de outros dados coletados. Dessa forma,

[11] `https://github.com/gobbisanches/ubisoldiers`

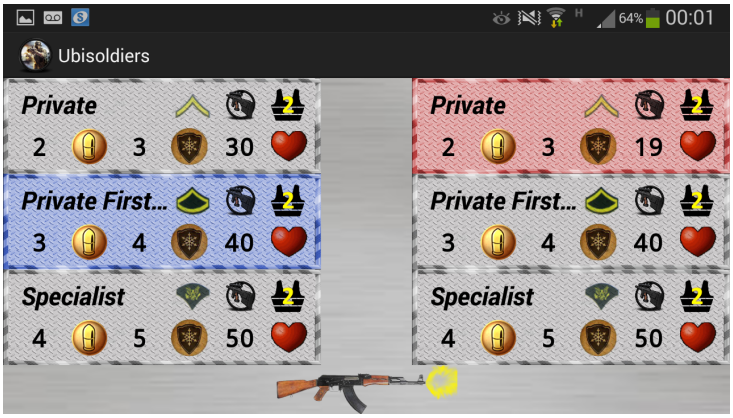


Figura 5.8: Combate ocorrendo no jogo *UbiSoldiers*.

um jogador que possua disponíveis tanto sua posição (obtida pelo GPS) como os dados de temperatura local, por exemplo, possui duas dimensões modificadoras de suas chances para aquisição de itens e unidades em uma busca.

Apesar das características de cada celular ser definida na sua fabricação, o UbiSoldier consiste em um único jogo que se adapta a cada uma destas configurações. Cada conjunto de sensores de cada celular possibilita uma composição de possibilidades distintas para a coleta de itens, diferenciando de outros *ARGs*. Mesmo assim, utilizando a localização como dado comum garante que zonas muito visitadas tenham suas chances minimizadas, forçando jogadores a buscarem zonas não exploradas a fim de obter itens mais raros.

5.3 Jogos Prototipados

Conforme dito anteriormente, quatro dentre doze jogos projetados chegaram ter seus protótipos implementados. Para isso, os desenvolvedores foram apresentados à plataforma de desenvolvimento *uOS* e receberam o prazo de um mês para desenvolverem seus títulos até a criação de um protótipo jogável. Dos quatro títulos produzidos, que são apresentados a seguir, dois fazem uso do *plugin Unity 3D*, um faz uso do motor *uImpala* e um faz uso estrito do *middleware uOS*.

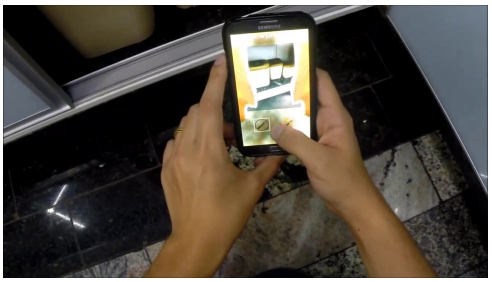
5.3.1 HarMegido

Em "*HarMegido*"[12] os jogadores interpretam o papel de anjos e demônios na eterna batalha entre o céu e o inferno. Divididos em times, os usuários se enfrentam através do encantamento de objetos presentes no ambiente. Quanto mais objetos (e por mais tempo) um time possuir, mais pontos esse time acumula. Jogadores podem também quebrar o encantamento dos objetos do time adversário, evitando assim que seus pontos sejam adquiridos. Ao final de dez minutos, o time com mais pontos vence a partida.

Para o encantamento dos objetos é utilizada a câmera do celular do jogador. Para tanto deve-se manter o enquadramento do alvo estável por cinco segundos, indicando o artefato desejado. Para se realizar um desencantamento, basta posicionar a câmera do celular em

[12] `https://github.com/vvgaming/HarMegido`

(a) Início de uma partida.



(b) Usuário encantando um objeto.

Figura 5.9: Dois momentos no jogo "*HarMegido*".

uma posição similar a utilizada durante seu encantamento pelo mesmo período de tempo. Foi observado durante os testes o surgimento de diversas estratégias em decorrência desta dinâmica. Por exemplo, usuários mais altos buscavam, obter ângulos mais difíceis de outros usuários reproduzirem. Outros usuários tentavam impedir que os encantamentos tivessem sucesso, desestabilizando seus adversários.

"*HarMegido*" pode ser caracterizado como um *ARG* onde o componente que conecta o mundo virtual ao mundo real é a câmera do celular. Desta forma objetos físicos ganham uma representação digital na forma dos encantamentos realizados. Este jogo fez uso do "*HTTP Plugin*" que permitiu as sessões do jogo se realizarem tanto em locais abertos ou fechados.

5.3.2 Ubimon

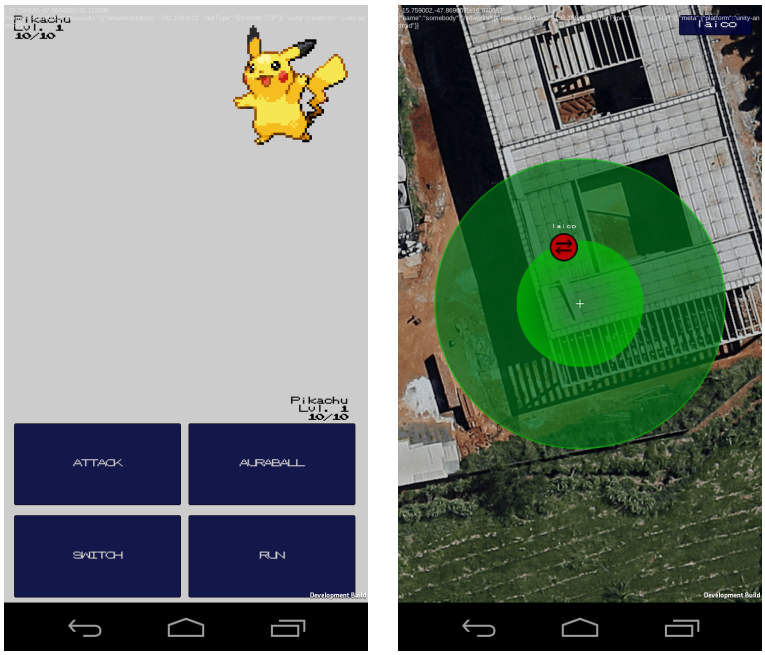
O jogo "*Ubimon*"[13] é inspirado pelo título comercial *Pokemon*™[14]. Nele, utilizando seu celular, o jogador deve capturar, carregar consigo e utilizar em batalhas pequenos monstros. Nesta versão, tais monstros são encontrados no ambiente a volta do usuário, sendo descobertos utilizando as informações de localização fornecidas pelo celular. De acordo com tais informações, são sorteados os *ubimons* que estarão disponíveis naquela região. O jogador pode optar por lutar contra eles, tentar capturá-los ou fugir (Figura 5.10(a)).

Cada *ubimon* possui uma representação gráfica cujo tamanho na tela é proporcional ao tamanho do monstro. Como as telas de dispositivos móveis possuem tamanhos distintos, um jogador não pode carregar consigo mais unidades que seu dispositivo comporta. Para evitar esta situação, o jogador pode utilizar uma das estações de armazenamento (Figura 5.10(b)), que são computadores públicos onde é possível deixar *ubimons* armazenados. Assim como os celulares, as estações possuem uma capacidade proporcional ao tamanho de sua tela.

"*Ubimon*" seria um *ARG* tradicional onde temos apenas o uso de localização para criar a relação do mundo real com o virtual. Porém o uso do tamanho da tela do celular torna o jogo adaptável a cada dispositivo e sua configuração, trazendo novos obstáculos para o jogador superar. Além disto, a existência das estações de armazenamento (telas públicas) não é apenas um facilitador para o limite de espaço, mas também mais um ponto

[13] `https://github.com/lhsantos/ubimon`

[14] Jogo produzido pela *Nintendo* e disponível para diversos consoles.



(a) Uma batalha contra um *ubimon* encontrado.

(b) Mapa de jogo apontando a localização de uma estação de armazenamento.

Figura 5.10: Três momentos no jogo "*Ubimon*".

de conexão com outros dispositivos. Este jogo fez uso tanto do "*HTTP Plugin*" para a comunicação como do "*Unity 3D Plugin*" como motor de jogo.

5.3.3 Vidi

"*Vidi*"[15] é uma adaptação do jogo de cartas *Dixit*[16] para o celular com a inclusão de elementos do ambiente. Sua mecânica consiste em cada jogador possuir um conjunto de cartas contendo desenhos abstratos e criativos. Cada participante inicia o jogo com um conjunto aleatório de cartas que são então utilizadas em trocas ou para desafiar outros jogadores. Pode-se também descartar uma carta em uma localização específica, permitindo que um outro usuário a encontre e tome sua posse.

Quando um jogador inicia um desafio, ele deve escolher uma carta e junto com ela fornecer uma palavra ou pequeno texto que a descreva (Figura 5.11). Usuários próximos podem aceitar aquele desafio e colocarem em jogo uma carta que se acredite ter correlação ao texto apresentado. Quando um número suficiente de participantes aceita o desafio, as cartas escolhidas por todos lhes são exibidas, incluindo a do desafiante. Cabe então, a cada desafiado escolher qual carta, dentre todas, possui maior chance de ser a proposta pelo desafiante. Quando o número de escolhas certas e erradas for equilibrado, o desafiante vence, levando consigo todas as cartas em jogo. Quando este equilíbrio não ocorre, as cartas em jogo são distribuídas entre os jogadores que acertaram a escolha. Caso ninguém tenha realizado a escolha correta, todos ficam com suas cartas originais.

[15] `https://bitbucket.org/lucasncv/vidi`

[16] Criado por Jean-Louis Roubira e publicado pela *Libellud*.

(a) Tela inicial do jogo. (b) Seleção de desafio.

Figura 5.11: Dois momentos no jogo "*Ubimon*".

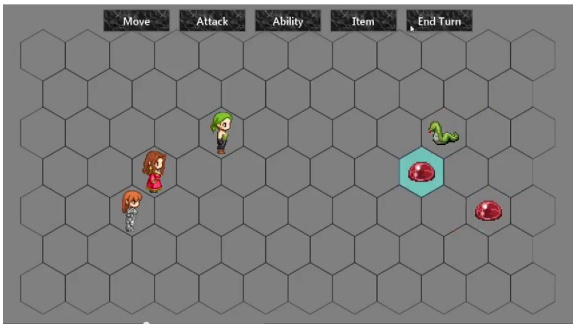
Vidi é um exemplo de *LBG* onde a informação de localização é responsável por conectar os elementos do jogo ao mundo real. Neste caso, posicionar os desafios no espaço onde eles foram lançados. Este jogo fez uso tanto do "*HTTP Plugin*" para a comunicação como do "*Unity 3D Plugin*" como motor de jogo.

5.3.4 uRPG

Em "*uRPG*"[17] o jogador, utilizando seu laptop, controla um grupo de heróis em uma terra cercada de magia e aventura. Para avançar no jogo, ele deve cumprir missões, criar e trocar itens, participar de batalhas dentre diversas outras tarefas possíveis (Figura 5.12(a)). O mundo do jogo é composto por continentes, onde cada um destes é representado pela rede *WiFi* a qual o jogador está conectado. Ao chegar em um continente, o jogador pode ver outras equipes (de outros jogadores), terras selvagens (onde ocorrem batalhas) e cidades. Cidades são na verdade computadores fixos do ambiente, onde o jogo está rodando.

É nas cidades que o jogador pode executar a maior parte das ações que o ajudam a progredir. É lá que novas habilidades são aprendidas, heróis são contratados para compor sua equipe e missões são aceitas. São nelas também que itens podem ser confeccionados, trocados, comprados e vendidos. A maior parte destas ações não são passíveis de serem concluídas no mesmo continente. Por exemplo, uma missão pode requisitar que o jogador visite um tipo de região selvagem que não se encontra próximo dele. Isso força o jogador a

[17] `https://github.com/LBNunes/uRPG`



(a) Batalha no jogo.

(b) Continente da rede "Billy mkII".

Figura 5.12: Dois momentos no jogo "*uRPG*".

visitar novos locais (com novas redes e dispositivos) para que avance com o jogo. Porém, para obter os espólios de uma missão, o jogador deve retornar à cidade onde a mesma foi aceita originalmente.

"*uRPG*" é um exemplo de jogo espontâneo onde as informações da rede sem fio, bem como os dispositivos lá presentes alteram a configuração do jogo. Isso obriga o jogador a buscar novas redes (continentes) e dispositivos (cidades) a fim de completar suas missões e assim avançar. Este jogo faz uso da *uImpala* na sua implementação.

5.4 uSect

O jogo *uSect*[18] é um exemplo de jogo aberto desenvolvido como parte deste trabalho. Nele o jogador é apresentado a um mundo novo cercado de pequenas criaturas que vivem no interior de todo dispositivo computacional. O papel do jogo, neste contexto, é atuar como uma janela que expõe esse universo desconhecido ao usuário. Cabe então ao usuário escolher que tipo de participação ele deseja ter. Sua interação pode ser nula, apenas um observador do desenrolar do ecossistema, ou ele pode agir sobre seus acontecimentos tanto de forma construtiva quanto destrutiva.

Neste mundo, existem dois tipos de criaturas (denominados por "*sects*") identificados por suas cores: azul ou laranja. Criaturas azuis possuem um comportamento pacífico. Passam seu tempo percorrendo o ambiente atrás de pedaços ("*bytes*") perdidos de memória. Eles se alimentam destes *bytes*, mantendo o ambiente sempre limpo de qualquer "sujeira" existente. Criaturas laranjas são agressivas quando com fome, caçando e atacando outros que estejam próximas a elas. Elas se alimentam das carcaças de *sects* mortos.

Todo *sect* possui uma quantidade de energia limitada. Caso se passe muito tempo sem se alimentar, ele virá a morrer. Outras ações possíveis às criaturas incluem o ataque e reprodução, sendo que ambos consomem energia extra de seus iniciadores. No caso de uma criatura atacar outra, caso bem sucedida, o dano causado será superior à energia gasta. Já no caso da cópula, caso ambos os *sects* concordem com a ação, um novo *sect* nasce. Este tem 50% de chances de herdar as características de um de seus progenitores.

Como cada dispositivo representa um ambiente distinto, suas características de *hardware* influenciam no funcionamento dos *sects* que ali habitam. Para esta versão do jogo

[18] `https://github.com/nuk/uSect/`

são utilizadas as informações de resolução de tela, velocidade de processamento e tamanho de memória. Porém, informações adicionais podem ser introduzidas em versões futuras, de acordo com metáforas que se julguem adequadas. A utilização destas características faz com que cada dispositivo ofereça uma experiência de jogo diferente, incentivando o jogador a buscar mais dispositivos para interagir.

A resolução da tela de um dispositivo determina o "espaço físico" que o ambiente possui. Como cada *sect* ocupa uma área bem definida, esse tamanho determina a população máxima que um local comporta. A velocidade do processador determina a fluidez desse espaço. Sendo assim, processadores mais capazes levam a ambientes com movimentos mais ágeis, enquanto aqueles mais limitados levam a uma dinâmica mais vagarosa e calma. Por fim, o tamanho de memória total determina as chances (a cada *frame*) de um *byte* da memória se perder no ambiente. Tais *bytes* são representados por pequenos pontos verdes, e, como dito anteriormente, servem de alimentos para as criaturas azuis. Desta forma, ambientes com pouca memória possuem menos fartura de nutrientes, enquanto o contrário ocorre com aqueles onde uma grande quantidade de memória se encontra disponível.

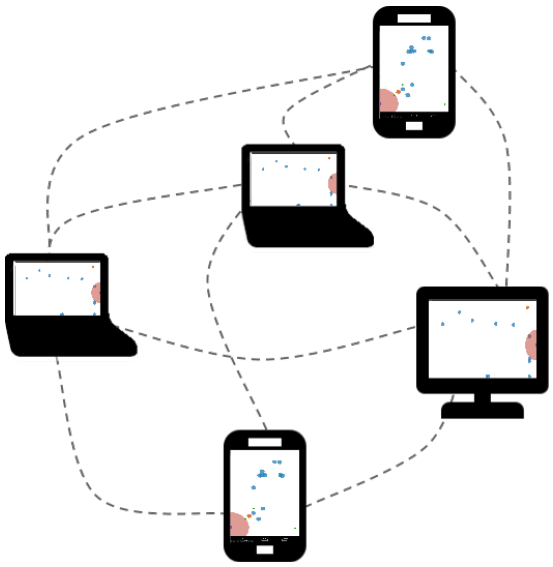


Figura 5.13: Representação das "pontes" entre ambientes *uSect*.

Uma das formas estabelecidas para que novos *sects* apareçam é a procriação entre duas criaturas. Outro método possível é por meio da migração entre dispositivos distintos. Isto se dá utilizando "pontes" (Figure 5.13) disponíveis apenas quando dois ou mais ambientes estão presentes no mesmo *smart space*. As "pontas" (vistas como portas) destas pontes são representadas através de retângulos magentas desenhados na borda da tela e indicam a conexão entre dois ambientes distintos. Caso um *sect* entre em uma destas áreas, ele é transportado para o ambiente na outra ponta. *Sects* não são cientes da existência destas pontes, podendo cair nelas apenas pelo acaso de seus movimentos.

Como um jogo aberto, o *uSect* permite que seus usuários interajam com o ambiente através de recursos compatíveis com os esperados pelo sistema. No caso, o jogo se utiliza da interação de pressionamento do botão principal de um cursor. Por padrão, esta ação é apenas interpretada através de recursos ligados a dispositivos pessoais, permitindo identificar qual jogador comandou a ação. Durante a execução, quando uma interação

ocorre, todas as portas das pontes ligadas ao dispositivo do usuário propagam um círculo de interação. Este círculo se expande e contrai, permitindo que se capture *sects* para seu ambiente ou descarte-os para outros. Como é utilizado o recurso de cursor para esta ação, caso outro equivalente esteja disponível no dispositivo, como um acelerômetro, este pode ser utilizado de forma transparente.

Da forma como é apresentado o jogo, seus comportamentos já permitem diversas interações. Porém, os comportamentos dos *sects* são limitados apenas àqueles providos nas criaturas de cor laranja (carnívoros) e azul (herbívoros). Como um jogo aberto, o *uSect* permite que seus usuários criem seus próprios comportamentos, expandindo as interações e mecânicas possíveis, através do uso de unidades de execução. A forma para se introduzir uma *EU* (*Execution Unity*) no jogo é bastante simples, bastando inserir o *script Lua* desejado em uma pasta observada pelo sistema. Ao identificar que um novo *script* foi introduzido, um *sect* correspondente é então criado no ambiente. Como essas criaturas podem se mover entre ambientes, bem como carregar seus comportamentos para seus descendentes, um novo comportamento pode se propagar entre ambientes distintos. Dessa forma, a criação de um usuário pode se espalhar por ambientes nunca visitados por seu autor.

A Listagem 5.1 apresenta um exemplo de como um novo tipo de *sect* pode ser criado utilizando o suporte provido. O ciclo de vida de cada criatura segue um modelo similar ao existente em jogos como *RoboCode* e *Light-Bot*, onde o método *update* (Linha 3) representa a atualização do estado e ações desejados. De forma complementar, os métodos *onEntered* (Linha 24) e *onLeft* (Linha 27) informam de mudanças ocorridas no ambiente. Usando os recursos do *Execution Driver* para a adição de métodos auxiliares, são fornecidas informações do ambiente (como no método *positionOf*) e ações aos *sects* (como nos métodos *move* e *attack*, nas Linhas 9 e 15, respectivamente).

5.5 Desempenho

Além da preocupação funcional da plataforma, provendo um conjunto de serviços que facilitem o trabalho do desenvolvedor, existe também a questão do desempenho apresentado pela mesma. Isto é importante, já que o uso dela impacta não apenas na experiência percebida pelo usuário, mas também em seu rendimento no próprio jogo [68]. Observando apenas o tempo de atraso entre a interação realizada pelo usuário e o retorno recebido podemos estabelecer como limiares aceitáveis tempos dentro de 100 ms[19] e tempos acima de 500 ms como inaceitáveis ao jogador. Na composição do atraso de uma mensagem existem três componentes envolvidos: a rede sendo utilizada como meio, a plataforma e o jogo (aplicação) sendo construída. Sendo os dois primeiros escolhas do desenvolvedor, porém fora de seu controle, é de seu interesse que estes adicionem a menor fatia possível no tempo de atraso existente, restando mais margem a ser utilizada pela lógica do próprio jogo.

A Figura 5.14 apresenta como cada componente contribui com o "atraso" em um jogo. O interesse do desenvolvedor é que o atraso introduzido pela combinação da plataforma e a rede seja mínimo, permitindo a execução do máximo de lógica do jogo dentro da janela estabelecida de 100 ms. Para avaliar este dois componentes (rede e plataforma), foram

[19]Onde abaixo de 50 ms não é perceptível mesmo por especialistas.

```
1  targetPosition = nil
2  knownSects = {}
3  function update()
4    checkTarget()
5    if (not (targetPosition == nil)) then
6      x = targetPosition['x'] - position['x']
7      y = targetPosition['y'] - position['y']
8      move(x,y)
9      attack()
10   end
11 end
12 function checkTarget()
13   for index, id in pairs(knownSects) do
14     if distance(id) < 100 then
15       targetPosition = positionOf(id)
16     end
17   end
18 end
19 function distance(id)
20   p = positionOf(id)
21   return math.abs(position['x']-p['x'])
22            + math.abs(position['y']-p['y'])
23 end
24 function onEntered(data)
25   table.insert(knownSects, data['id'])
26 end
27 function onLeft(data)
28   for index, id in pairs(knownSects) do
29     if id == data['id'] then
30       table.remove(knownSects, index)
31     end
32   end
```

Listing 5.1: Exemplo de um *sect* onívoro e caçador.

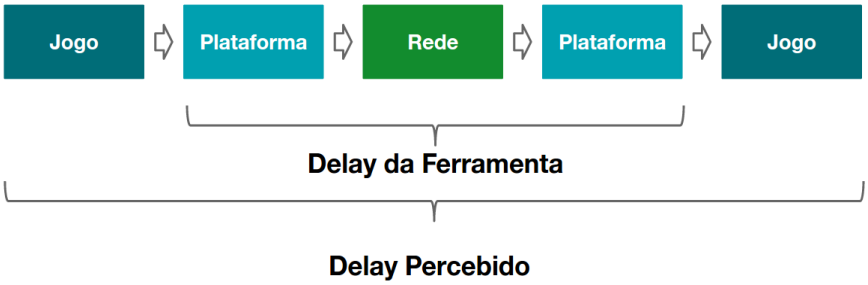


Figura 5.14: Estrutura dos componentes que influenciam no atraso percebido pelo jogador.

realizados dois experimentos. O primeiro sem a influencia da rede, buscando analisar apenas a influência da plataforma. Já o segundo contando com ambos, avaliando o desempenho em um ambiente real. Ambos os testes foram realizados utilizando *hardware* e redes reais não isoladas para os fins destes testes, representando uma situação mais próxima de onde o jogo irá ser executado. Os resultados são discutidos a seguir e estão detalhados no Apêndice B.

5.5.1 Atraso da Plataforma

Para avaliar o atraso impelido somente pela plataforma, foram realizados testes com diversas configurações de *hardware*. Nestes foi utilizado o tempo de conclusão de chamadas de serviço síncronas com a variação do número de parâmetros e o tamanho destes (em bytes). Este experimento replica o mesmo realizado na primeira análise de desempenho do *middleware uOS* [20] e permite avaliar o impacto das alterações implementadas durante a construção da plataforma. Os tempos obtidos compreendem do momento da realização de uma chamada de serviço até a sua recepção no *driver* requisitado em um ambiente restrito a apenas um dispositivo. Definindo assim qual o atraso provado apenas pela plataforma, sem a influência da rede.

O menor tempo encontrado foi de 0,007 ms enquanto o maior de 0,880 ms, colocando o impacto da plataforma como irrisório dentro do limiar estabelecido. Estes tempos se mostram bem inferiores aqueles encontrados em plataformas para jogos como o *STF/ARMS* [37], com tempos de 150 ms[20], ou mesmo de plataformas de *ubicomp* como o *PolyChrome* [9], com tempos de 10 ms. Quando comparado com os dados obtidos na mesma configuração utilizada para os testes da versão 2.0 do *middleware uOS*, os resultados também são melhores, caindo de 1,935 ms [20] para 0,031 ms. As duas razões que mais contribuem para esta melhoria está no uso de "mapas *hash*" e a prevalência de conexões. No primeiro, devido a retirada da carga de gestão imposta por uma ferramenta de banco de dados. Já no segundo, pela remoção do custo de comunicação após a primeira troca de mensagens.

5.5.2 Atraso Total

Para avaliar o custo de transmissão de uma mensagem utilizando a rede foi estabelecido uma topologia de rede sem fio *802.11g* onde cada nó foi avaliado com relação ao tempo de

[20]Este valor foi obtido em testes realizados em ambiente simulado em uma máquina com processador Pentium 4 3.6 GHz.

transmissão de uma mensagem de tamanho fixo. Com isso é possível avaliar não apenas o tempo de recebimento determinando o atraso entre uma ação do jogador (ou jogo) e sua resposta subsequente no outro dispositivo.

Dentre os resultados obtidos o pior tempo de recebimento foi de 71,315 ms e o melhor de 50,374 ms, o que coloca o impacto da conjunto entre 50% e 70% do limiar não perceptível ao usuário. Estes valores também demonstram que o atraso da plataforma na comunicação é bastante insignificante perante o atraso introduzido pela rede, sendo responsável 1% deste impacto.

Capítulo 6

Conclusão e Trabalhos Futuros

A busca por novas formas de se entreter os usuários sempre foi um motivador constante na indústria de jogos eletrônicos. Isso fica evidente a medida que meios tradicionais como consoles e computadores pessoais cada vez mais dão espaço a novos meios de interação. As novas mídias já são responsáveis por mais da metade do faturamento da indústria de jogos desde o ano de 2012 [8]. Tal tendência é suportada pela crescente difusão de dispositivos móveis interconectados a uma grande parcela da população mundial[1]. Soma-se ainda que a diversidade de dispositivos à disposição desses usuários apenas cresce, com a presença cada vez mais facilitada de sensores biométricos, *smart-watches*, óculos de realidade aumentada dentre diversos outros [2].

Dentro deste cenário, os jogos ubíquos se apresentam como uma das formas de se utilizar as capacidades presentes nos dispositivos, criando novas possibilidades de diversão. Isso fica claro no caso de títulos de mercado como *Ingress* e *GeoCaching* que possuem seu número de jogadores na casa dos milhões. Porém, o nível de inovação com relação ao *game design* permanece reduzido porque a maior parte dos jogos desenvolvidos na academia tem seu foco na tecnologia utilizada em detrimento da diversão por ela proporcionada [63]. Vale destacar que entre os *ubigames* houve surgimento de novos gêneros, embora estes tenham se concentrado, em grande parte, em duas vertentes: os *ARGs* e *LBGs*[3]. Além disso, as plataformas construídas para suportar o desenvolvimento de jogos não atendem aos desafios comuns à *ubicomp*:

- **Heterogeneidade**: capacidade de integrar plataformas de hardware e software distintas, bem como diferentes tecnologias de comunicação.
- **Mobilidade**: permitir a mobilidade de dispositivos, usuários e aplicações pelo ambiente, assim como entre ambientes distintos.
- **Integração Espontânea**: incorporação de novos dispositivos, bem como novos tipos de funcionalidades em tempo de execução.

[1] Hoje a penetração da internet já alcança 3 bilhões de pessoas, com grande parte disso sendo realizada através das mais de 7 bilhões de assinaturas de telefonia móvel disponíveis no globo terrestre.[88]

[2] Apenas para citar alguns destaques dentre estes novos dispositivos, temos o *Pebble*, *Google Glass*, *FitBit* e *Microsoft Kinect*

[3] Dentro do levantamento feito para este trabalho, os dois gêneros de jogos representam aproximadamente metade dos títulos encontrados no Apêndice A

- **Sensibilidade ao Contexto**: possibilitar que as aplicações tenham acesso às informações do ambiente e também às alterações ocorridas em tempo real, permitindo que ações adequadas sejam tomadas.

Acrescente-se a essas características o fato das plataformas atuais possuírem seu foco apenas na interconexão entre os dispositivos, não prestando um suporte especializado no desenvolvimento de jogos.

Considerando esses desafios, as duas principais contribuições deste trabalho se constituem na definição de jogos ubíquos reconfiguráveis e na plataforma *uOS* para jogos ubíquos. A primeira apresenta como um ambiente ubíquo pode ser melhor utilizado, favorecendo a criação de novos tipos de jogos. Já a última fornece um conjunto de ferramentas que apoiam o desenvolvimento de jogos ubíquos, facilitando não só a integração de dispositivos, mas também o desenvolvimento dos jogos que os utilizem. As seções que seguem discutem com mais detalhe as contribuições deste trabalho.

6.1 Jogos Ubíquos Reconfiguráveis

Os jogos ubíquos reconfiguráveis buscam utilizar uma das características mais marcantes da *ubicomp*, a incorporação espontânea e dinâmica de dispositivos, para criar novos tipos de jogos. Isso pode ser observado nos dois gêneros decorrentes - Jogos Espontâneos e Jogos Abertos - aqui propostos. Estes dois gêneros mostram como a autoadaptação de um jogo à configuração do ambiente permite uma experiência de jogo que sempre se modifica, dessa forma incentivando o jogador a buscar novos ambientes a fim de alcançar uma maior variabilidade de sessões de jogo.

Complementar a isto, a classificação em três níveis apresenta não só exemplos de jogos que já possuem características de reconfigurabilidade, mas também formas de como esta pode ocorrer. Com base nos exemplos, é possível observar que, apesar de tímida, a inclusão de aspectos dos jogos ubíquos está ocorrendo no mercado tradicional de jogos eletrônicos como naqueles baseados em consoles e em computadores pessoais. Cada um dos níveis apresenta como a reconfiguração pode auxiliar a expandir a experiência de jogo, partindo das menos impactantes até as mais complexas, estas servindo de guia para a criação de jogos que se adaptem ao ambiente que o cerca.

O conjunto de meios de interação apresentados provê uma base de conhecimento a projetistas de jogos na criação de seus títulos, possibilitando a escolha de quais dispositivos, técnicas e tecnologias serão mais adequados para a representação, no mundo real, daquilo que se deseja no universo do jogo. A eles se acrescenta o compêndio de jogos presente no Apêndice A, que serve como referência aos títulos existentes de jogos ubíquos até o presente momento.

Por fim, os onze títulos desenvolvidos no âmbito desta tese demonstram de que modo as mudanças do ambiente poderiam ser embarcadas dentro do projeto de jogos por meio da criação de experiências que mudam de acordo com a configuração do ambiente.

6.2 Plataforma *uOS* para jogos ubíquos

A plataforma *uOS* disponibiliza ferramentas que dão suporte ao desenvolvimento de jogos ubíquos, endereçando os desafios que estes enfrentam. A tabela 6.1 reapresenta a tabela 2.1, incluindo agora a plataforma uOS para efeito de comparação.

	Heterogeneidade		Integração Espontânea	Mobilidade	Sensibilidade à Contexto	Componentes de Jogo
	Múltiplas Redes	Múltiplas Plataformas	Novos Tipos	Novos Dispositivos	Interfaces Definidas	
MUPE	Não	Não	Não	Não	Sim	Não
PSD	Não	Sim	Não	Sim	Não	Não
fAARS	Não	Não	Não	Sim	Não	Não
GameWork	Não	Não	Não	Sim	Não	Não
uOS	**Sim**	**Sim**	**Sim**	**Sim**	**Sim**	**Sim**

Tabela 6.1: Comparativo dos desafios suportados pela plataforma *uOS* em relação as demais.

Dois componentes presentes no *middleware uOS* permitem que as questões de heterogeneidade sejam endereçadas: o uso de *plugins* de rede e o conjunto de protocolos de comunicação *uP*. O uso de plugins permite que tipos de comunicação distintos sejam utilizados por um mesmo dispositivo. Tal suporte se mostrou bastante flexível, possibilitando que ao longo deste trabalho fosse desenvolvido o *plugin HTTP*, complementar aos já existentes *plugins* para *Bluetooth* e *Sockets TCP/UDP*. Com relação a este, foi também adicionada a opção de se utilizar um radar baseado em anúncios em *multicast*, mais eficientes que os radares por varredura. Isso visa melhorar o suporte já existente à integração de novos dispositivos ao *smart space*.

A especificação das interfaces para meios de interação definida na hierarquia de recursos auxilia tanto no desenvolvimento de recursos reutilizáveis como sua integração aos jogos. Essas interfaces servem de linguagem comum entre os jogos que as utilizem, permitindo acesso, sem ambiguidade, às informações e capacidades do ambiente. Com base na relação de equivalência provida pela *DSOA*, é possível ainda que se criem novas interfaces - referentes a novos tipos - de modo que jogos existentes sejam compatíveis sem a necessidade de adaptação.

O motor de jogo *uImpala* fornece uma *API* multiplataforma de suporte à construção de jogos. Este tipo de recurso não é observado em nenhuma das outras plataformas para *ubigames*. É um motor que incentiva a aplicação de padrões de desenvolvimento e outras boas práticas já estabelecidas na indústria de jogos. Além disso, mecanismos para tratar de forma transparente dispositivos de entrada e saída, bem como conteúdos e mídias são fornecidos, favorecendo o reuso de código e simplificando o projeto de jogos que utilizem a plataforma. Complementarmente, tem-se ainda o *plugin* para o motor de jogo *Unity 3D*, que permite que jogos desenvolvidos nesta plataforma tenham acesso a recursos em um *smart space uOS*.

O *Execution Driver* expande as capacidades do ambiente inteligente ao permitir que não apenas jogos, como também aplicações em geral, tenham seu comportamento modificado em tempo de execução. Ao contrário de outras ferramentas de apoio à mobilidade de

código [11, 12, 35, 85], o uOS não apenas suporta as três estratégias típicas da mobilidade, mas o implementa na modalidade multiplataforma através do uso de *scripts Lua*.

A validação da Palaforma *uOS* foi realizada através do desenvolvimento de sete jogos que utilizaram seus recursos. Em especial pode-se destacar o *uSect*, um exemplo de jogo aberto que explora não apenas a adaptação às capacidades voláteis do ambiente, mas também a mobilidade de código de forma a alterar o comportamento do jogo sob o controle de seus usuários.

Por fim a plataforma apresentou resultados satisfatórios nos testes de execução. O impacto em termos de atraso manteve-se entre 50% a 70% do limiar de percepção do usuário em ambiente de rede sem fio, não interferindo de forma perceptível na jogabilidade.

6.3 Trabalhos Futuros

O trabalho aqui apresentado atende aos desafios inicialmente propostos, expondo uma nova forma de utilizar os recursos computacionais no projeto de jogos e auxiliando no tratamento das questões de heterogeneidade, integração espontânea, mobilidade e sensibilidade ao contexto. Apesar disso, restam ainda outras questões não respondidas neste trabalho além de evoluções por ele apoiadas. A seguir, são destacadas algumas destas.

6.3.1 Integração entre ambientes

A plataforma *uOS* permite que dispositivos em um ambiente possam se comunicar entre si e, complementarmente, dispositivos e aplicações possam trafegar em ambientes distintos. Apesar disso, um ambiente ainda desconhece os demais, sejam estes distantes entre si ou próximos. Possibilitar a troca de informações entre ambientes faria uma aplicação enxergar além das capacidades existentes no espaço onde ela se localiza. Consequentemente, propiciaria meios para o usuário vasculhar ambientes tanto adjacentes quanto distantes em busca de interações que lhe fossem interessantes, sem a necessidade de se locomover fisicamente.

6.3.2 Base de recursos

Durante o desenvolvimento deste trabalho foram desenvolvidos alguns recursos de uso comum - como teclados e cursores - além da definição das interfaces de interação para um extenso conjunto de capacidades. Expandir tais recursos, não só pelo desenvolvimento de novas implementações como disponibilizando-as através de uma base de código livre, contribuirá para incrementar seu reuso. Desta forma, desenvolvedores que buscam utilizar um meio de interação podem fazer uso daqueles já existentes. Adicionalmente, a criação de novas interfaces e implementações pode ser disponibilizada, expandindo a lista aqui apresentada.

6.3.3 Integração com outras plataformas

O *plugin* para o motor *Unity 3D* apresenta como outras plataformas podem ser incorporadas em um ambiente *uOS* de maneira transparente. Apesar de permitir a comunicação

entre aplicações e recursos de maneira bidirecional, este *plugin* não possui todas as funcionalidades disponíveis pela implementação corrente do *middleware*. O desenvolvimento de uma biblioteca que opere de maneira similar, utilizando ferramentas distintas (como *SDL/C++*[4]) aumentaria ainda mais o alcance da plataforma.

6.3.4 Aprimoramento do *plugin HTTP*

O *plugin HTTP* existente permite a comunicação entre dispositivos de maneira transparente através da nuvem computacional. Na sua versão atual, o *plugin* permite a existência de apenas um servidor, o que se caracteriza como um gargalo na comunicação. A implementação de uma lista de servidores, bem como sua atualização e descoberta de forma dinâmica, reduz esse impacto em jogos de maior escala. Além disto, dispositivos presentes em uma rede local podem ser utilizados como *proxies*, conectando ambientes distintos de forma similar à plataforma *STF/ARMS* [37], expandindo desse modo o acesso a recursos entre *smart spaces* distintos.

6.3.5 Evolução do *Execution Driver*

Apesar de possuir o suporte à transmissão de agentes - objetos Java - utilizando a infraestrutura provido pela *JVM*, este recurso se limita a àrvores de dependências não muito complexas. Esta restrição impede que unidades de execução mais sofisticadas como *drivers* ou mesmo aplicações completas sejam transferidas. Tal capacidade expandiria de forma significativa as possibilidades de mobilidade e adaptação no ambiente. Sua incorporação ao *middleware* envolveria o desafio de compatibilizar a carga de dependências da máquina virtual bem como otimizar sua transferência, evitando duplicidades. Apesar de não suportar diretamente esta tarefa, a plataforma *OSGi* [5, 60] possui ferramentas que podem auxilia-lá.

6.3.6 Suporte à renderização remota

Dentre os diversos tipos de estímulos disponíveis aplicáveis em um jogo, os visuais permanecem os de maior visibilidade ao usuário. Devido a sua importância, seria de interesse a plataforma fornecer meios de que fossem utilizados de maneira transparente e efetiva. A transmissão de fluxos de vídeo apresentam um alto consumo de banda, onerando bastante a rede de comunicação. Por esta razão que plataformas como o *MUPE* [56] fazem uso de descritores, delegando a renderização para o cliente. Uma outra possibilidade seria a utilização de mensagens que encapsulem as chamadas da *API OpenGL*, permitindo a representação de forma dinâmica e flexível de estímulos visuais em recursos de vídeo.

6.3.7 Controle de contexto utilizando ontologias

Jogos como "*Save the Princess*" [64] observam as mudanças no ambiente através do casamento de expressões regulares contra tuplas de dados coletados. Já a plataforma *PSD* [51] realiza essa tarefa através de *queries* de contexto denominadas *CIQs*. No *middleware uOS*, tal controle se dá através do uso dos serviços assíncronos (eventos). O suporte dado

[4]Do inglês *Simple DirectMedia Layer*. `https://www.libsdl.org/`

pelo *middleware* para a construção de ontologias [26] pode ser utilizado para a criação e manutenção de uma base de conhecimento, expandindo os tipos de inferências que os jogos podem realizar sobre os dados disponíveis.

6.3.8 Realização de testes mais avançados

Durante o processo de desenvolvimento dos títulos apresentados utilizando a plataforma, não foram apresentadas dificuldades quanto ao seu uso. Apesar de ter sido realizada, uma análise quantitativa existe uma falta de dados comparativos de desempenho, carga e facilidade acerca de outras plataformas, tanto para jogos como de aplicações ubíquas em geral.

6.3.9 Desenvolvimento de outros títulos

O desenvolvimento de jogos é o objetivo deste trabalho e das ferramentas nele desenvolvidas. Portanto, o desenvolvimento de mais títulos que não façam uso apenas do conceito de jogos ubíquos reconfiguráveis e da plataforma é desejado. Esta pode ser utilizada não apenas nos gêneros aqui apresentados, mas em diversos outros.

Apêndice A

Levantamento de jogos ubíquos

Neste apêndice são apresentados um conjunto de jogos (bem como tecnologias de suporte) encontrados tanto na academia como no mercado durante a pesquisa realizada para este trabalho.

A.1 TMP -The Malthusian Paradox

Este *ARG* [36] foi um experimento de narrativa realizado simultaneamente em diversas cidades da Europa. Seu objetivo consistiu na análise como a narrativa poderia ser manipulada para se alcançar um maior engajamento dos jogadores. O jogo teve como base quatro universidades situadas em cidades distintas. Em seu início, estudantes foram convidados a uma palestra sobre os perigos dos alimentos geneticamente modificados. Nesta palestra, um ator interpretando o doutor Baxter tentava convencer os alunos dos malefícios dessa técnica. Porém, no meio da palestra, o doutor é sequestrado por uma organização chamada TFT. Em meio a confusão, os presentes são convidados a ajudar a desmascarar a organização e resgatar o doutor. Para isso, os jogadores devem se juntar a uma entidade rebelde chamada AMBER.

Durante 3 meses, os jogadores receberam conteúdos visuais e textuais contendo uma atriz interpretando a sobrinha do doutor. Tarefas foram requisitadas aos jogadores para que estes descobrissem mais da história. Entre estas tarefas havia a busca por objetos nas cidades, encontrar pessoas específicas, responder SMSs e outros desafios. Por fim, o jogo contou com quase 300 jogadores participando pelo site e fórum do grupo rebelde.

A.2 FreshUp

"*Fresh Up*" [97] tem por objetivo avaliar o engajamento de alunos universitários em um jogo que os ensina sobre a vida no campus. Neste jogo, calouros eram convidados a realizar tarefas ligadas a serviços providos pelo campus e a cidade onde a universidade está localizada. Tais tarefas envolviam desde a coleta de informações com pessoas específicas até a presença em locais e horários preestabelecidos. O jogo se utiliza de uma interface HTML5, acessando as informações que cada tipo de dispositivo permite e adaptando sua base de tarefas de acordo. A iniciativa alcançou um alto grau de engajamento bem como da absorção das informações passadas.

A.3 Augmented-TCG

Em "*Augumented TCG*" [75] o objetivo está em avaliar como recursos computacionais podem aumentar o engajamento de jogadores adultos de jogos de troca de cartas (*TCGs*). De acordo com o estudo, adultos tendem a jogar menos este gênero de entretenimento tendo em vista a distância de outros jogadores e a impessoalidade das interações online. Para solucionar esta questão, o jogo se utiliza de técnicas de realidade aumentada e narrativa para reproduzir um ambiente próximo ao desenho animado Yu-gi-oh, motivador do jogo de cartas.

Neste ambiente cada jogador tem a sua frente uma mesa onde estão projetadas as cartas de seu oponente à frente das suas. As carta do jogador é identificada através de um sensor Kinect posicionado acima da mesa. Além disto, os movimentos de cada jogador é capturado através de um sensor Kinect que os reproduz a um avatar projetado a frente do jogador. Complementarmente, uma tela auxiliar apresenta o avatar de um personagem que busca auxiliar com dicas de como se jogar.

Através dos experimentos foi visto que apesar de interessante, o aparato não aumenta o engajamento. O uso de avatares não facilita a imersão no ambiente do desenho. Além disto, a tela auxiliar remove muito da imersão do jogo através de suas interrupções.

A.4 MuseUs

O *MuseUs* [28] é um "jogo sério" com o objetivo de auxiliar visitantes de um museu na criação de sua própria narrativa. O usuário é convidado a montar seu roteiro escolhendo as obras que deseja visualizar. Ao longo do jogo, frases provocam o visitante a encontrar obras e meditar sobre elas. Conforme o jogador encontra as peças correspondentes, sua jornada pelo museu é completada.

Cada obra é identificada através de um código *QR*, identificado pela câmera do celular do visitante. Neste, é exibido um ambiente virtual tridimensional do espaço criado pela narrativa do jogador. Pequenos quadros são exibidos conforme são encontrados, ao passo que dicas são exibidas com relação aos itens pendentes a se obter.

A.5 Pervasive World Search

Em "*Pervasive World Search*" [89] o jogador é desafiado a descobrir a palavra escondida. Para isso ele deve buscar elementos ao seu redor que o auxiliem na tarefa através de três sensores de seu celular. Primeiramente, pode-se utilizar as letras presentes nas sete cores básicas. Para isso, o jogador deve usar a câmera do seu celular para identificar elementos com a cor desejada. Uma segunda fonte de informações são os nomes de dispositivos *Bluetooth* próximos. Por fim, pode-se utilizar as letras presentes no *SSID* de redes *WiFi* no alcance do celular. Este jogo exemplifica como a lista de características ubíquas pode auxiliar no projeto de jogos ubíquos.

A.6 Outbreak: Safety First

O jogo "*Outbreak: Safety First*" [46] é um protótipo construído para se exercitar a plataforma fAARS. Este jogo é um *treasure hunt* onde estudantes de medicina são ensinados sobre os riscos epidemiológicos em um hospital. Os jogadores devem visitar os leitos dos pacientes e realizar os procedimentos adequados a cada paciente. Para tal, cada local é identificado por códigos *QR*. Ao chegar em um local, os jogadores recebiam as informações acerca dos pacientes e a lista de opções de ações a serem realizadas. De acordo com as escolhas dos jogadores e sua localização, eles podiam ser infectados e transmitir a condição para outros pacientes e jogadores.

A.7 iFitQuest

Este *exergame* [59] tem por objetivo capturar a essência dos jogos causais e utilizá-la para manter a motivação dos jogadores por um maior período de tempo. É o jogador que determina seus desafios e como deseja progredir no jogo. Em sua primeira versão, foram disponibilizados oito minijogos divididos em três grupos:

- "*Collect the Coins*" e "*Visit the Fields*" são jogos que impulsionam o jogador a visitar locais e coletar objetos virtuais lá localizados.
- "*Escape the Wolf*", ''*Return the Sheep*" e "*Follow the Chicken*" requerem mais esforço físico simulando uma fuga ou a caça à animais.
- "*Mistery Games* 1, 2 e 3" convidam o jogador a correr o mais longe possível em um determinado espaço de tempo resolvendo mistérios de cada historia.

  O jogo foi experimentado por um grupo de doze crianças, porém os efeitos sociais e individuais deste grupo não permitiram avaliar se os jogos realmente aumentaram a motivação de seus jogadores.

A.8 U-Theather - Smash the Beehive

Este jogo [51] consiste em um evento que ocorre em um ginásio onde uma grande tela pública está disponível. São convidados todos os jogadores a se posicionar de frente a tela onde é exibida uma colmeia cercada por um enxame de abelhas. Para permanecer no jogo, os participantes devem se manter imóveis. Seus movimentos são detectados por sensores em seus corpos. Quando o enxame se dissipa, os jogadores devem atacar a colmeia o mais rápido possível. Aquele que o fizer mais rapidamente, ganha a partida. Este jogo é um dos protótipos da plataforma *PSD* e exemplifica como sensores podem ser incorporados através de celulares como dispositivos intermediários.

A.9 Swan Boat

Este *exergame* [51] desafia a coordenação dos esforços de dois jogadores. Divididos em duplas, cada participante deve escolher um equipamento a ser usado para interagir, o qual, por sua vez, fornece um dado distinto: a esteira e a bicicleta ergométrica informam

sua velocidade, enquanto o bambolê informa o número de rotações por unidade de tempo. Cada jogador então representa um lado do barco em uma corrida pelo rio. A velocidade com que cada jogador impulsiona seu equipamento representa a velocidade no seu lado do barco. Desta forma, cada dupla deve coordenar suas ações para desviar dos obstáculos e alcançar a linha de chegada primeiro. Diferentes escolhas de equipamentos acabam por criar níveis distintos de dificuldade no jogo. Este é um dos protótipos usados pela plataforma *PSD*.

A.10 SwordFight

Este jogo [98] se utiliza de uma técnica de localização baseada no som emitido e capturado por celulares. Tal técnica possui baixo consumo energético e computacional ao mesmo tempo que mantém um bom tempo de resposta. Cada jogador é um espadachim, e seu objetivo é ser o único sobrevivente da batalha. Para atacar, o jogador necessita apenas tocar a tela de seu celular. O ataque tem um alcance máximo de 20 cm e possui um custo energético. Quanto maior o período que o jogador manter a tela pressionada, até um máximo de 4 segundos, maior a energia gasta. Os jogadores podem recuperar sua energia ficando longe da batalha por um tempo. É necessário calcular bem a distância de seus ataques e balancear a energia gasta entre os ataques e fugas.

A.11 Treasure

Este jogo [43] tem por objetivo permitir que seus jogadores introduzam os elementos do ambiente e definam como eles vão influenciar na mecânica do jogo. Para isso, antes de cada sessão de jogo um "autor" deve cadastrar os objetos do ambiente que farão parte do jogo. Para cada objeto é determinada uma descrição (ou história), uma dica e uma ação a ele relacionada. Para a identificação destes elementos são utilizados um equipamento dotado de sensores ultrassônicos, um projetor, uma câmera e uma base rotacionável. Os elementos jogáveis são identificados através do uso de etiquetas ultrassônicas sobre os quais são projetadas imagens e textos relativos as ações determinadas. Ao longo da sessão, jogadores devem percorrer o ambiente buscando os objetos corretos de acordo com as dicas e ações estabelecidas.

A.12 Candy Castle

"Candy Castle" [81] é um jogo pervasivo focado na promoção da saúde infantil, em especial crianças que sofrem de diabetes. O jogo se baseia em um título que o antecede, porém analógico em seus jogadores a consciência de estar sempre controlando o nível de glicose no sangue. No jogo, cada criança é responsável por proteger seu castelo dos ataques vindos das "forças obscuras". Para tal, deve-se construir e manter uma muralha em torno do castelo, cuja construção exige que o usuário faça a medição de sua glicose em locais (identificados por GPS) em torno do seu castelo, formando assim um perímetro de proteção. Caso se passe muito tempo sem se expandir a muralha, estas começam a ser atacadas pelas "forças obscuras" podendo ser destruídas. O jogo também fornece os dados

coletados ao médico responsável, ajudando a controlar o histórico da criança. Alertas são gerados quando os dados informados se encontram em níveis perigosos, o que favorece ações rápidas e efetivas. Todavia, os dados são registrados pelos próprios usuários, apenas, sem controle externo de sua veracidade.

A.13 1-2-3

"1-2-3" é um protótipo de utilização do *framework GameWork* [82] inspirado no jogo *GeoCaching*. Nele o jogador pode criar missões geolocalizadas, para que seus amigos as cumpram. Estas missões envolvem visitar um ou mais lugares e em cada um destes satisfazer aos eventos (como questões a serem respondidas) determinados. As missões são desenhadas pelo desafiante através da elaboração de fluxos (grafos), possibilitando diferentes desdobramentos.

A.14 Pervasive Pairs

Pervasive Pairs, ou apenas P^2, é um jogo da memória geolocalizado baseado no *framework GameWork* [82]. Neste jogo, pessoas em diferentes locais devem percorrer distâncias no mundo real de forma a "virar" as cartas do jogo. Tanto as imagens usadas nas cartas, quanto sua posição no mapa podem ser estabelecidas por seus jogadores.

A.15 Mobilis Xhunt

Este jogo é uma variação geolocalizada do título "Scotland Yard", no qual os jogadores devem capturar um bandido fugindo pelas ruas de Londres. Porém, nesta versão, a caçada é mapeada para o mundo real utilizando a plataforma *Mobilis* [79]. No lugar dos pontos fixos do tabuleiro, são utilizadas estações de transporte público como pontos de ônibus e estações de metrô. A cada turno os jogadores só podem ir fisicamente para pontos que sejam adjacentes aos que estão posicionados. Além disso, só é permitido o uso de transporte público nesta locomoção. Um turno tem seu fim, quando todos os jogadores registraram sua nova posição. A cada cinco rodadas é revelada a posição atual do jogador que interpreta o bandido. Caso um jogador chegue a uma localização onde o bandido se encontra, este é capturado e o jogo se encerra.

A.16 Mister X ® Mobile

Este jogo é uma variação geolocalizada do título "Scotland Yard", onde os jogadores devem capturar um bandido fugindo pelas ruas de Londres. Porém, nesta versão, a caçada é mapeada para o mundo real utilizando a plataforma *Mobilis* [79]. Diferente da versão original, esta versão ocorre em tempo real (sem o uso de turnos). O objetivo dos jogadores é chegar próximos ao bandido para capturá-lo. Para se evitar que a caçada se estenda por um espaço muito grande, é delimitado um raio onde o jogo deve ocorrer. A posição do bandido é revelada de tempos em tempos ou caso ele saia do raio estipulado. Além

disto, são espalhados pelo mapa itens que auxiliam os jogadores na caçada, aumentando o dinamismo do jogo.

A.17 MobilisLocPairs

Este jogo [58] é uma variação do jogo da memória (*Pairs*) utilizando a plataforma *Mobilis*. Neste, as cartas a serem viradas são etiquetas *QR* localizadas em diferentes pontos de um prédio.

A.18 Geocaching in the city

Consiste em uma variação do jogo *GeoCaching* utilizando o suporte do sistema *WiFly* [92]. O sistema permite a troca de serviços entre diferentes pontos de acesso sem fio (*APs*). No jogo, cada *AP* representa uma *geocache* onde estão disponíveis serviços para registro dos usuários que ele visitaram. O aplicativo é executado em celulares e consulta *caches* próximos a localização do usuário informando quando este está próximo o suficiente para se registrar. Para tanto podem ser verificados a força do sinal da rede, a detecção de uma etiqueta *QR* ou uma palavra chave.

A.19 Lunch Time

"Lunch Time" [66] se autodenomina como um Jogo Multijogador Casual Lento. Seu objetivo é auxiliar na mudança de hábitos alimentares através da combinação de metas, influência social e mecanismos de recompensa. O jogo consiste em um grupo de usuários que são desafiados diariamente a realizar dez escolhas distintas acerca de suas refeições. Cada escolha é composta por três opções, as quais a mais adequada assegura 10 pontos, 5 pontos para a neutra e 2 para a menos adequada. Os jogadores podem realizar suas escolhas a qualquer momento do dia, ao final do qual são enviadas mensagens com informações acerca das escolhas realizadas, bem como o *ranking* dos membros do grupo.

A.20 AbueParty

Este jogo [62] tem por objetivo auxiliar idosos na prevenção (e compensação) de seu declínio na percepção, cognição e condicionamento motor. Ele é composto por um jogo de tabuleiro apresentado em uma tela pública, da qual para se avançar entre as casas, é preciso superar desafios (mini jogos) que estimulam as capacidades dos usuários. Estes jogos incluem:

- Desafios Musicais: O jogador deve cantar uma música ou adivinhar qual está sendo tocada.
- Desafios Artísticos: O jogador deve atuar ou desenhar um objeto.
- Desafios de Coordenação: O jogador deve construir (desafio motor) algo ou realizar a comparação entre objetos.

A.21 Hoodies and Barrels

"*Hoodies and Barrels*" [7] tem por objetivo ajudar crianças a aprenderem sobre matemática brincando com as noções de dimensão e distância em duas variações do jogo de pega-pega. Para isso são utilizados dois elementos de jogo: jaquetas e barris. As jaquetas são artefatos vestíveis utilizados pelas crianças. Nelas encontram-se etiquetas RFID e um mostrador de *LED* que se comunica através de *ZigBee* com uma central. Já os barris possuem uma antena *RFID* que permite estimar a distância das crianças e informar isso para a central. Em uma primeira modalidade do jogo, o professor estabelece desafios a serem resolvidos pelas crianças. Estes desafios envolvem problemas lógicos que são registrados antes da sessão de jogo. Um exemplo de limitação seria o de todas as crianças se esconderem atrás de um objeto com mais de 50 metros cúbicos. Ao inicio da partida, o jogador recebe a restrição e deve então encontrar seus amigos. Em uma segunda modalidade, as crianças se escondem e definem dicas de sua localização. O jogador "caçador" deve então seguir as dicas, ativadas por proximidade, para encontrar seus amigos.

A.22 Adaptative Target Shooting Bike Game

Este *exergame* [80] busca desafiar o jogador fisicamente, se adaptando de acordo com o nível de esforço físico alcançado pelo jogador. Para intergir, é utilizada uma bicicleta ergométrica posicionada de frente a uma tela de projeção. Nesta é exibido o cenário de uma ilha onde três alvos, distâncias e posições distintas estão localizados. O objetivo do usuário é atingir estes alvos arremessando cocos. Para pontuar, os três alvos devem ser derrubados em menos de dez segundos. Para isso, a direção no eixo horizontal fica oscilante enquanto o jogador deve determinar a força de arremesso de acordo com a velocidade sendo alcançada. Para arremessar, utiliza-se um botão no guidão da bicicleta. Com base nos resultados do jogador são ajustados quatro fatores de dificuldade do jogo: a resistência da bicicleta, a velocidade necessária para se lançar o coco, a frequência de oscilação do eixo horizontal e a distância dos alvos exibidos.

A.23 Outdoor Pico Safari

Em *Pico Safari* [45], o objetivo do jogador é encontrar e capturar animais virtuais (chamados de picos) bem como itens utilizando apenas seu celular. Usando o *framework fARPLAY* tal o jogo pode ser jogado tanto em ambientes abertos, utilizando a localização obtida pelo *GPS*, ou em ambientes fechados, utilizando etiquetas *QR*. Sendo a localização a única informação utilizada para se interagir, picos e itens são coletados quando se está a menos de 50 metros de distância.

A.24 Campus Misteries

Em *Campus Misteries* [45] o jogador é convidados a conhecer mais sobre os mistérios envolvendo a história da universidade de Alberta. Nesta "caça a relíquias" cabe ao jogador vagar pelo campus da universidade em busca de enigmas e as pistas que o auxiliem na resolução. Usando recursos de realidade aumentada disponíveis no celular através do

framework fAR-PLAY o jogador visualiza fantasmas e outras entidades que o auxiliam a progredir em sua missão.

A.25 Blowtooth

Este jogo [55] busca descontrair o ambiente sério e agitado dos aeroportos através de missões lúdicas em que não jogadores participam sem estarem cientes disto. O objetivo de cada jogador é conseguir "traficar" itens através dos pontos de segurança do aeroporto usando outras pessoas como "carregadores". Para isso o jogo busca ao redor os identificadores *bluetooth* presentes e armazena eles como os carregadores. Após uma espera de dez minutos, o jogador deve então buscar estes carregadores de forma a obter os itens traficados. Como os identificadores nem sempre fornecem informações de quem os possui, cabe ao jogador encontrá-los em meio a tantos passantes.

A.26 Treasure Hunt NFC

Neste jogo de caça ao tesouro [39] diversas etiquetas *NFC* são espalhadas pelo ambiente físico de forma a estabelecer o "tabuleiro" da sessão. Estas etiquetas marcam os alvos que os jogadores devem alcançar em cada estágio dentro do tempo e na ordem determinada. Jogadores mais rápidos são bonificados com itens e bônus de pontuação enquanto os mais lentos podem receber penalidades. Os jogadores visualizam o jogo na tela de seus celulares, onde é possível ver o tabuleiro mas não os alvos correntes. Jogadores que se encontrem no meio do caminho podem se confrontar utilizando sua pontuação e itens encontrados.

A.27 iCat (Chess)

Este projeto [25] busca analisar como a emoção de um oponente virtual pode influenciar na reação dos jogadores. No jogo construído temos um "gato robô" que serve de adversário a crianças em um jogo de xadrez. Ao longo da partida o gato apresenta-se como um companheiro de jogo, de forma a auxiliar no aprendizado do enxadrista. Por esta razão, as emoções expressas tentam reproduzir como seriam as reações do oponente, e não mimetizar as da criança em jogo. Desta forma, expressões de tristeza são apresentadas quando o gato perde (ou está perdendo), e de alegria caso contrário.

A.28 Sanningen om Marika

Este *ARG* [6] realizado na Suécia consistiu em uma ação em diversas mídias levando parte de seus participantes a crer que se tratava de uma ação verídica. O jogo teve início com uma série de TV exibida por um mês na TV do país. Nesta eram expostos os conflitos acerca de relacionamentos, em especial o de Adrijhanna em busca sua amiga de infância Maria, que ela suspeita ter sido sequestrada por uma ordem secreta chamada Ordo Serpentis. Juntamente com a série foi disposto um site chamado Conspirare, onde informações sobre o drama de Adrijhanna eram compartilhadas bem como fóruns e *chats* para que outros pudessem participar. Durante a ação, alguns eram convidados a tomar parte

na organização Entropia, de forma a combater a Ordo Serpentis. Quem era selecionado devia superar desafios, tarefas e missões de forma a subir de posição na organização.

A.29 King of Location

Este "*capture the flag*" [87], construído utilizando o *framework FRAP*, simula a batalha de dois times onde quem conquistar mais territórios vence a partida. Cada ponto é definido como um local real pré-definido pelo jogo. Um time conquista um local quando este possui um número maior de integrantes naquele local que o time adversário. Isso faz com que cada time tenha que equilibrar a conquista de novos pontos com a defesa dos já conquistados. Vence o time que tiver mais pontos sob seu domínio ao fim da partida.

A.30 Battle Bots

Neste jogo [10], crianças controlam tanques robôs usando o movimento do seu corpo. Estes tanques podem se movimentar, bem como atirar uns nos outros usando *LEDs*. Para controlá-los, as crianças fazem uso de um casaco bem como luvas contendo sensores de movimento. Movimentos como, inclinar para a frente move o robô para a frente, enquanto para trás o movimento inverso. Quanto mais intenso o movimento, mais rápido ocorre a movimentação do robô. As luvas são utilizadas tanto para mirar a a arma laser (LED) do robô bem como atirar utilizando um botão presente.

A.31 FeedBall

A *FeedBall* [10] é uma bola incrementada com um sensor de aceleração e um acelerômetro que permitem determinar a direção e precisão do passe realizado. Usando esta informação a bola apresenta, através de *LEDs* qual a "qualidade" da jogada realizada pela criança, fornecendo um *feedback* imediato da ação. Isso é utilizado em diversos jogos como o jogo de cinco passos. Neste, dois times devem marcar pontos no mesmo gol. Porém, para que a pontuação seja válida, cinco passes devem ser realizados. A cada passe realizado com sucesso a bola aumenta a intensidade de seus *LEDs*, indicando quanto falta para que seja possível marcar o gol. Quando se erra um passe, ou a bola fique parada por dez segundos, esta volta ao estado inicial. O time com mais pontos vence a partida.

A.32 LEDTube, Color Flare e MultiModal Mixer

Este experimento [10] avaliou como diferentes sensores e atuadores podem influenciar na emergência de brincadeiras por crianças. Em um primeiro experimento foi criado o *LED-Tube*, um cilindro contendo apenas um acelerômetro e *LEDs RGB*. Ao se rolar, ou agitar o bastão, suas cores eram alteradas. Posteriormente foi criado o *Color Flare*, adicionando ao tubo um sensor infravermelho que permitia que a cor fosse transferida entre bastões. Por fim, o *MultiModal Mixer* acrescentou a emissão de sons (ao agitar) e vibração (no recebimento de uma cor por infravermelho). A cada iteração do dispositivo era avaliada,

junto a um grupo de crianças, a diversidade de brincadeiras que surgiam. Com o acréscimo de novas interações, a emergência foi aumentada durante os experimentos.

A.33 UbiBall

A *UbiBall* [34] é uma bola que possui um acelerômetro, emissores de luz e som. Utilizando-a, temos um jogo que ocorre em duas partes: uma real e uma virtual. Na primeira parte as crianças participantes trocam passes em uma pista onde o desafio é superar obstáculos com o mínimo de faltas possíveis. A bola informa quando um passe é bem sucedido ou não através de sons e padrões luminosos. Duas crianças começam o percurso em um ponto do campo no qual devem percorrer até a cesta que se encontra no final. O detentor da posse de bola não pode se mover, sob risco de penalidade. Caso a criança saia do percurso ou esbarre em um obstáculo, esta deve retornar ao início.

Na segunda parte os jogadores carregam os dados coletados pela bola para um computador onde os resultados do jogo físico serão refletidos como bônus para serem usados no jogo virtual. Neste, novamente o jogador deve percorrer uma série de obstáculos no menor tempo possível, porém agora no mundo virtual. O jogo virtual consiste um *side scroller* que espelha o jogo original, porém com os bônus ganhos.

A.34 Moving Monk

Aqui [4] o jogador interpreta o papel de monges que buscam conquistar os templos inimigos através da força de seus mantras. Para se ganhar a partida, um dos participantes deve conquistar todos os templos adversários. Cada templo é representado por uma estação de jogo (um dispositivo *SunSPOT*) posicionado em um local específico determinado pelo alcance da rede do dispositivo. Os mantras são representados através de gestos, sendo que cada templo tem um movimento único que o identifica. Para capturar os movimentos são utilizados os acelerômetros do *SunSPOT*.

A.35 Assassin Apprentice

Este jogo [4] simula a desafiadora vida em uma guilda de assassinos onde a traição pode vir a qualquer momento. Um dos jogadores assume o papel de mestre e a ele é possível matar qualquer um de seus aprendizes com um gesto. Já os aprendizes, para derrotar seu mestre, devem realizar todos ao mesmo tempo o gesto. Tais gestos são realizados utilizando um dispositivo *SunSPOT* e devem estar próximos ao jogador que se deseja inflingir o golpe.

A.36 Day of the Figurine (DoF)

Este "jogo evento" [40] faz uso de mensagens de texto *SMS* para permitir que os jogadores interajam com os acontecimentos. Aqui o jogador é convidado a viver um dia na história de uma pequena cidade, onde cada dia no mundo real representa uma hora no universo do jogo. No início do jogo, a cidade se apresenta como pacata mas ao decorrer da história

diversos eventos vão ocorrendo. A cada momento o jogador pode informar à qual região da cidade ele deseja se deslocar. Desta forma, o jogador só possui a visão do que ocorre onde ele se encontrava. A localização de cada jogador é representada em uma maquete física que pode ser acompanhada através de um site.

A.37 Love City

Neste jogo [40], três cidades físicas têm diversas de suas localidades mapeadas em uma única cidade virtual. Os jogadores devem ir a esses locais para que suas representações virtuais possam se mover. Conforme eles andam por este mundo virtual, podem encontrar e interagir com outros jogadores. É através destes encontros que cada jogador tem a possibilidade de encontrar seu "verdadeiro amor"ao enviar-lhe uma mensagem anônima. Caso ela seja aceita, ele ganha o jogo, caso contrário, ele é eliminado. Toda interação pode ocorrer através de mensagens SMS, interpretada por um servidor, ou através de um aplicativo no celular.

A.38 Professor Tanda

Neste jogo [40] o personagem Professor Tanda convida o jogador (duas a três vezes ao dia) a participar em pequenas missões e questionários sobre os cuidados com o ambiente. Nestes eventos, são desafiados seus hábitos (como medir quanto de água se consome no banho) ou questionados seus conhecimentos (múltipla escolha ou abertas) sobre o meio ambiente. Além dos momentos predeterminados, o jogador pode invocar o professor a qualquer momento.

A.39 Mobi Missions

Neste *ARG* [40] os jogadores participam e criam missões que os convidam a visitar locais espalhados pela cidade. A localização é identificada com base na célula de telefonia móvel ao qual o celular usado está associado. Ao completar uma missão, o jogador deve tirar uma foto comprovando seu sucesso. Jogadores podem votar nas melhores missões e comentar o que tem de positivo ou negativo em cada uma delas. Através de um site é possível acompanhar o estado das missões e quem as completou.

A.40 IFloorQuest

Utilizando o dispositivo de interação *IGameFloor* [42], foi construído o jogo *IFloorQuest*. Este dispositivo consiste em um piso interativo onde, além das projeções realizadas, é possível identificar a silhueta das pessoas presentes. Operando, então, tanto como um dispositivo de entrada como de saída. O jogo consiste em uma gincana onde os jogadores devem alinhar blocos de pedra sob as respostas que julgam mais corretas.

A.41 UbiSettlers

Este jogo [49] de estratégia em tempo real coloca jogador responsável por administrar uma ilha representada por seu dispositivo pessoal. Nesta é possível erguer edificações, coletar recursos e combater ameaças. Quando jogadores se encontram na mesma rede, é possível realizar trocas de maneira distribuída. Observando os nós que fazem parte de cada troca, jogadores em redes distintas são conectados, bonificando aqueles que facilitam este comércio.

A.42 Drink Some Beer

A missão do jogador aqui [56] é encontrar a jarra de cerveja no mundo do jogo e bebê-la elevando seu nível alcoólico. Quando seu nível estiver alto o suficiente, ele pode utilizá-lo para se teleportar para outros locais do jogo e busca de outras jarras. A localização no jogo é determinada através da trangulação *WiFi* do jogador dentro do prédio da universidade.

A.43 Snow War

Neste jogo [56], dois times competem em uma guerra de bolas de neve. Cada um dos times deve estabelece suas bases em um dos locais dentro do prédio da universidade. Quando um jogador encontra a localização de uma base inimiga, a ele é permitido realizar um ataque a mesma. Este ataque ocorre na forma de lançamentos de bolas de neve contra uma parede de obstáculos. Quando uma base está sob ataque, o jogador do time a que esta pertence é notificado. Este é então invocado a defendê-la, evitando que as bolas de neve atinjam a parede. Caso isso ocorra, cabe a ele reparar a parede para atrasar o progresso da invasão inimiga.

A.44 Hunters

Vários jogadores participam de uma caçada [56] pelo prédio da universidade. Quando dois jogadores se encontram no mesmo local, eles devem se desafiar em uma corrida na bicicleta ergométrica.

A.45 Treasure Hunt

Nesta caçada ao tesouro [56] os jogadores devem se mover entre os ambientes do prédio em busca de itens escondidos. Porém, cada movimento deve ser bem planejado, já que só é possível se mover dentro de determinados períodos de tempo. Caso esta restrição não seja respeitada, o jogador é punido. Outros elementos estratégicos incluem a colocação de armadilhas nos locais a fim de atrasar seus oponentes.

A.46 Pre-Emptive Strike

Neste jogo [56], cada time deve procurar as partes de uma bomba que se encontram espalhadas pelos diversos locais do prédio. Quando todos os pedaços da bomba forem coletados, ela deve ser instalada na base inimiga para ser detonada, ganhando-se assim jogo. Partes da bomba podem ser roubadas de jogadores do time inimigo ao se passar por eles no mesmo local.

A.47 Speed Biking

Este jogo [56] desafia o controle e resistência aeróbica do jogador. Em cada estágio é apresentado um nível de rotação a ser alcançado e mantido pelo jogador utilizando uma bicicleta ergométrica. Para se concluir cada estágio, deve-se manter o nível almejado pelo período de tempo estabelecido. A faixa e o período tornam cada estágio mais difícil.

A.48 Magic Mushroom Race

A cada turno deste jogo [56] um cogumelo é posicionado em um dos locais do prédio da universidade. O primeiro jogador a chegar ao local detém sua posse. Neste momento, um novo cogumelo é posicionado em outro lugar aleatório do prédio. Um jogador pode apostar seus cogumelos contra outros jogadores de forma a acumular mais cogumelos (ou perdê-los). O jogador que possuir mais cogumelos ao final da partida, ganha.

A.49 Cannon Game

Consiste em um jogo [56] de batalha entre dois jogadores através do controle de um canhão com o objetivo de atingir (e destruir) o canhão inimigo. Em turnos alternados, cada jogador pode ajustar o ângulo e a força do tiro a ser disparado. Porém, a cada turno a direção e força do vento é alterada, tornando cada lançamento distinto um do outro. A informação do vento é obtida com base nos dados climáticos do local onde se está jogando.

A.50 Save the Princess!

O jogo [64] conta a conhecida fábula da princesa aprisionada em um castelo. Para salvá-la, os jogadores devem enfrentar o cavaleiro negro que a mantém em cárcere. Para isso eles devem encontrar os ingredientes necessários no preparo da poção que os ajudará no combate. A cada jogador tem um papel específico no time, garantindo habilidades específicas para cada situação. Cada local ou objeto é identificado através de um sensor capaz de se comunicar com dispositivos na vizinhança. Os jogadores devem percorrer os locais em busca dos itens necessários para produzir a poção. Alguns locais estão guardados por monstros a serem enfrentados. Alguns destes itens podem ser usados durante as batalhas ou combinados para formar outros itens.

A.51 Tycoon

Este jogo [65] coloca o usuário no papel de um explorador no velho oeste. Locais da cidade são mapeados (através de células de telefonia móvel) em minas (fontes de recursos como ouro, prata e cobre) ou corretoras. Cabe aos jogadores andar pela cidade a fim de descobrir quais são esses locais. Quando próximos a eles, interação ocorre. Um jogador começa a coletar recursos quando próximo à região de uma mina. Cada corretora permite a compra de propriedades presentes na cidade, sendo que estas só podem ser adquiridas uma única vez. O objetivo do jogador é conquistar o maior número de propriedades disponíveis, se tornando o dono da cidade.

A.52 Hitchers

Este jogo [33] busca exercitar o uso de torres de celular como forma de localização em aplicações móveis. Sua mecânica convida os jogadores criarem caroneiros virtuais que desejem trafegar por estas células. Quando um caroneiro é criado, a ele é dado um nome e um destino. De forma similar, quando um caroneiro é deixado em um local, o jogador deve dar um nome aquele local. A cada jogador é permitido carregar apenas um caroneiro por vez. O objetivo do jogo é compartilhar suas experiências com caroneiros encontrados e acompanhar quais locais os seus caroneiros alcançam.

A.53 ShootBall

Cada partida desse jogo [84] consiste em dois times de três pessoas competindo com uma bola que contém em si um acelerômetro e um sensor de proximidade. A arena é composta por quatro telas projetadas nas paredes de uma sala onde duas telas correspondem ao gol de cada time e as demais às ações de deslocamento e mudança. Em cada gol são exibidos blocos que devem ser destruídos utilizando a bola, quando todos os blocos de um gol forem destruídos, a partida se encerra. Ao se atingir a tela de deslocamento, a posição das telas é deslocada a esquerda. Já a tela de mudança realiza a permuta entre as telas de gol, invertendo suas posições.

A.54 Cron

O objetivo deste jogo [57] é desenhar um determinado símbolo no mapa do jogo através da conquista (marcação) de locais físicos. O primeiro time a completar o desenho, ganha o jogo. Para conquistar um local, os jogadores devem ir até o local portando seus *PDAs*. Lá serão apresentadas informações (áudio, vídeo e texto) sobre aquela posição, bem como outras passíveis de serem conquistadas. Caso um time esteja sozinho em um local, basta permanecer neste local por um tempo determinado para marcá-lo. Caso exista um outro time no local, inicia-se um duelo. O duelo pode ocorrer através de perguntas sobre a história do jogo ou através de um jogo de bola virtual. O time vencedor marca o local. Após marcar um local, ele fica impedido de ser marcado por outro time por um período, evitando duelos contínuos.

A.55 Uncle Roy All Arround You

O jogo "*Uncle Roy All Around you*" [13] (Figura 2.1) consiste em um "jogo evento". Suas sessões tiveram como cenário uma das praças da cidade de Londres na Inglaterra, em 2003. Voluntários eram convidados a participar da caçada ao pedido do "'tio *Roy*" e para isso lhes era dado um *PDA* contendo o mapa da região. Para alcançar o objetivo, os jogadores deviam seguir uma série de pistas que os levavam a locais espalhados pelas imediações. Os jogadores contavam com a ajuda de outros participantes online, que poderiam contribuir com mensagens de texto ou áudio. Nessa atividade se observou como a interação entre jogadores compartilhando a mesma experiência, embora sob diferentes pontos de vista, ocorria. Juntamente, também se verificou como os usuários reagiam a um sistema de localização onde eles mesmos informavam suas posições (não coletada por nenhum tipo de sensor).

A.56 Can you see me now?

O Conceito do *CYSMN* [14] é baseado em uma perseguição entre jogadores virtuais e três corredores pelas ruas da cidade. Estes utilizam com *PDAs* com capacidade de comunicação *WIFI* e *GPS*. Jogadores online utilizam um modelo tridimensional para interagir com o jogo. Nele, devem tentar fugir dos corredores. A velocidade de movimento virtual é limitada, a fim de garantir a competitividade entre os jogadores. No modelo virtual é possível apenas se locomover pelas ruas da cidade, sendo possível ver outros participantes (online e corredores) e trocar mensagens com eles. O objetivo é evitar que os jogadores no mundo real cheguem a menos de cinco metros de um jogador virtual. Caso isso ocorra, este sai do jogo. A pontuação é determinada pelo tempo que um usuário durou dentro do jogo.

A.57 FantasyA

Criado como uma forma de exercitar o uso do boneco SenToy [67], o jogo utiliza a representação das emoções dos usuários como forma de interação. FantasyA [70] é um mundo guiado pelos sonhos de quatro deuses que, apesar de aprisionados em pedras (chamadas *KemY's*) usam os sonhos para impor suas vontades sob seus habitantes. Guiado pelos deuses, os *IdenjaYeh* construíram fortalezas que abrigam os clãs. Nestes locais, magos de cada clã são convocados a treinar na arte do *AlkemHYe* de controle elementar. Cada jogador de escolher a qual dos clas ele fará parte. Os jogadores interagem com o mundo se movendo, através da manipulação de gemas e interagindo com outros agentes através de duelos e trocas. Toda interação utiliza o boneco SenToy que permite ações de movimento e de representação de emoções distintas. Os movimentos estão restritos a andar, pegar itens e parar o personagem. Já as emoções são representadas são seis: felicidade, tristeza, raiva, desgosto, surpresa e medo

A.58 Smart Playing Cards

Este [74] é uma automação do jogo de cartas *Whist* que utiliza um baralho tradicional de 52 cartas por duas duplas de jogadores. De início cada jogador recebe 13 cartas enquanto uma é posicionada ao centro da mesa indicando a cor do "trunfo". A cada rodada, os jogadores devem posicionar suas cartas na mesa seguindo a estratégia que melhor lhes couber. Sendo este um jogo com regras complexas de serem absorvidas por um iniciante, o objetivo do sistema é auxiliar no seu aprendizado através do apoio computacional. Cada carta possui um marcador *RFID* próprio, permitindo que o jogador seja informado em seu *PDA* quando uma jogada realizada foi boa ou ruim de forma imediata. De forma complementar, uma tela pública indica de quem é a vez, a pontuação e se as jogadas são ou não válidas.

A.59 Touch-Space

Touch Space [27] se passa em uma sala de realidade aumentada, onde dois jogadores colaboram para salvar a princesa aprisionada pela bruxa. Para isto, três estágios devem ser superados, cada um com um nível distinto de imersão.

No primeiro estágio, deve-se encontrar os pedaços do mapa que indica a localização do castelo da bruxa. Para isso, os jogadores devem se deslocar pela sala dividida em quadrantes contendo caixas. Estes quadrantes (e suas caixas) podem possuir armadilhas, que custam pontos ao jogador dentro da partida. Para se identificar qual o item dentro de cada caixa, são utilizadas etiquetas *QR*. Estas também permitem a sobreposição de representações virtuais utilizando realidade aumentada.

No segundo estágio, os jogadores devem utilizar seus aviões, representados por uma "varinha" de realidade aumentada para derrotar a bruxa. Cada jogador controla a projeção de seu avião, atirando na bruxa e desviando de suas magias de fogo. O som dos elementos presentes é representado de forma a fornecer maior imersão, permitindo aos jogadores encontrarem onde está seu colega e a bruxa de forma mais simples.

Ao derrotar a bruxa, os jogadores entram no castelo. Neste estágio é apresentada uma imersão tridimensional onde deve-se encontrar a princesa seguindo o som de sua voz pelo labirinto de corredores.

A.60 Movement Smash Game

Este jogo [47] é um experimento da aplicação de um chão tátil (*Active Floor*) em jogos. Sendo uma variação do tradicional "*Whack a mole*", este jogo apresenta um plano contendo nove quadrantes mapeados nos quadrantes do piso usado como entrada do jogo. A cada momento são exibidos na tela itens (podendo ser bonificações ou punições) que devem ser coletados pelo usuário ao trocar de posição.

A.61 Movement Quake

Este [47] é uma adaptação do jogo de tiro em primeira pessoa *Quake* para o uso de um piso táctil. A locomoção do jogo é feita utilizando os quadrantes do piso, permitindo a

indicação de qual direção se deseja locomover. A frequência de passadas no local indica a velocidade do movimento bem como o ato de pular indica esta ação no jogo. Por fim, um sensor de condutividade atrelado ao pulso do jogador reconhece quando a mão se fecha em punho, indicando que se deseja atirar.

A.62 AR-Quake

Esta [86] é uma adaptação do jogo de tiro em primeira pessoa *Quake* para o uso de realidade aumentada através de *HMDs* (*Head Mounted Displays*). A posição dos jogadores é obtida através de *GPS* ligados a computadores pessoais nas mochilas usadas pelos jogadores. O mapa utilizado corresponde ao campus da universidade, em específico parte do departamento de computação. As ações disponíveis aos jogadores incluíam apenas a movimentação e tiro, sem incluir o pular. Para atirar e recarregar foram usados botões presentes em luvas usadas pelos jogadores.

A.63 Nevermind

Nevermind[1] é um jogo de terror em primeira pessoa que desafia o jogador a vencer uma série de quebra cabeças e transpor labirintos. Este jogo não apenas faz uso de uma realidade virtual imersiva (usando o *Oculus Rift*), mas também se utiliza de sensores externos, como de batimento cardíaco, para ajustar o funcionamento do jogo. Conforme o jogador fica mais agitado, o jogo se ajusta para ficar mais difícil, forçando o jogador a se acalmar para prosseguir.

A.64 Niantic Project

O gênero conhecido como *ARGs* ("*Alternate Reality Games*" do inglês Jogos de Realidade Aumentada) propõem a criação de mundos virtuais que usam o mundo real como seu tabuleiro. O jogo Ingress[2] se enquadra nesta categoria. Em sua história uma estranha energia está sendo detectada em diversos pontos do globo, esta originada de portais espalhados pelo mundo. Os jogadores devem escolher entre duas facções: a resistência, que acredita que tal energia é maligna e deve ser restringida; e os iluminados, que acreditam que esta energia deve ser estudada e explorada. Os jogadores então devem criar e capturar portais pelo mundo, estabelecendo suas estratégias em cada local. De forma colaborativa, o jogo também permite que seus participantes participem de fóruns e encontros onde a história do mesmo é cocriada por aqueles que dele fazem parte. Tendo sido lançado ao final de 2012, e ainda em andamento, estima-se um número de jogadores próximo de 7 Milhões [3].

[1] `http://www.nevermindgame.com/`
[2] `http:://www.ingress.com`
[3] `https://play.google.com/store/apps/details?id=com.nianticproject.ingress`

A.65 Geo Caching

Geocaching[4] é um *ARG* cujo objetivo é coletar recipientes (denominados de *caches*) escondidos pelo mundo. *Caches* dos mais diversos tipos e tamanhos (de cápsulas a caixas de munição) são escondidas por seus jogadores em locais estratégicos, onde não possam ser vistas por não jogadores (conhecidos como *mugles*). O desafio daqueles que criam as *caches* é aumentar o nível de desafio, encontrando lugares mais difíceis, camuflando ou adicionando diversos estágios para se completar.

A.66 LandLord Game

O jogo *Landlord*[5] é uma espécie de banco imobiliário. Usando as informações de locais disponíveis pelo *Foursquare*[6] o jogador pode comprar e aprimorar suas locações. Outros jogadores devem pagar uma taxa de aluguel quando visita uma locação que não lhes pertence.

A.67 Game of Cones

o jogo *Game of Cones* [7] consistiu em uma ação de marketing entre a equipe do serviço de localização *Foursquare* e a produtora *HBO*. Neste jogo, participantes presentes em Nova Iorque foram convidados a visitar o maior número de sorveterias. Ao final, a região que mais possuísse visitas ganharia o título de "Rei dos Cones".

A.68 Banco Imobiliário

A empresa Estrela, detentora dos direitos do jogo *Banco imobiliário* lançou uma versão do mesmo na forma de um jogo de realidade alternada. [8]. Neste é possível comprar locações (usando dinheiro virtual ou real) e receber aluguel.

A.69 Chrome Experiments

A fim de demonstrar as funcionalidades de integração presentes entre o navegador *Google Chrome* e os dispositivos Android foram criados diversos experimentos que permitem que tanto computadores desktop como celulares e tablets sejam usados de diversas maneiras. Dentre estes experimentos 3 se destacam:

[4] `https://www.geocaching.com/`
[5] `http://www.landlordgame.com/`
[6] `http://www.foursquare.com/`
[7] `http://gameofcones.foursquare.com/`
[8] `https://play.google.com/store/apps/details?id=br.com.pontomobi.estreladigital.bancoimobiliario&hl=en`

A.69.1 Super Sync Sport

Em *Super Sync Sport*[9] o jogador é convidado a participar de uma olimpíada. Composto por três mini jogos (corrida, ciclismo e natação), o navegador do celular é utilizado como controle, enquanto o navegador de uma tela pública serve de mostrador da partida.

A.69.2 World Wide Maze

World Wide Maze[10] transforma qualquer site em um labirinto tridimensional. Utilizando o giroscópio do celular, o jogador deve controlar uma bola através deste labirinto, levando-a até o final.

A.69.3 Kick with google

Este jogo[11] foi uma ação promocional relativa à copa do mundo de 2014 no Brasil. O jogador novamente utiliza seu celular como controle para uma partida sendo executada no navegador do computador pessoal. Três modalidades estão disponíveis: drible infinito, pênalti e chute mais forte.

A.70 Motion Tennis Cast

Este jogo de tênis[12] utiliza o recurso de expelhar a tela de celulares android para TVs, propiciando o uso do celular como sensor de movimento em um jogo de tênis. Após calibrar o aparelho é possível jogá-lo contra uma IA ou mesmo contra outros jogadores online. Além disso, diversos dispositivos podem se usados tanto como entrada (como *smart watches*) e saída (como *chromecasts*, *apple tvs* e outras TVs inteligentes).

A.71 Grand Theft Auto: iFruit

O jogo *iFruit* para celulares android[13] faz uma conexão entre um jogo casual de treinamento de cachorro com o jogo de console *GTA-V*. As ações conquistadas neste jogo são refletidas no console de maneira automática, conectando tudo em um mesmo universo.

A.72 Pokemon Dream World

A iniciativa "*Pokemon Dream World*"[14] desenvolvida pela *Nintendo* visa integrar diversos títulos diferentes da franquia *Pokemon* em uma experiência conjunta. Seu título principal disponível ao console portátil "*Nintendo DS*" é chamado de "*Pokemon Black and White*". Este convida seus jogadores a capturar e treinar monstros conhecidos como *pokemons*.

[9] `http://chrome.com/supersyncsports/`
[10] `http://chrome.com/maze/`
[11] `https://kickwithchrome.withgoogle.com`
[12] `http://www.motiontennis.tv/`
[13] `https://play.google.com/store/apps/details?id=com.rockstargames.ifruit`
[14] `http://www.pokemon-gl.com/`

Seja cuidando destes ou utilizando eles em batalhas contra outros treinadores. Em um navegador *web* se tem acesso a um conjunto de jogos casuais. Estes permitem aumentar o grau de amizade ou habilidade que os monstros possuem junto a seu dono. O resultado destes jogos se reflete no título principal, expandindo as consequências das ações e conectando suas narrativas. De forma complementar, o dispositivo "*Pokewalker*" permite acrescentar interações distintas ao jogador permitindo que este leve seu *pokemon* para um passeio. Tal artefato opera como um pedômetro, onde quanto mais longe se vá, mais experiência é acumulada pelo monstro selecionado. Soma-se ainda que o dispositivo permite a interação com outros treinadores que se encontre pelo caminho. Hoje essas experiências estão sendo transportadas para os novos títulos da franquia através da iniciativa *"Poke Bank"*[15].

A.73 Tom Clancy's The Division

Neste jogo[16] da produtora *Ubisoft* o jogador faz parte de uma divisão do governo responsável por remediar um surto de uma doença que assola os *EUA*. Estando disponível para os consoles de última geração, este jogo permite que quando experimentado em dispositivos distintos seu comportamento também mude, mantendo o mesmo universo. Quando no console, o jogador tem acesso a uma mecânica de tiro em primeira pessoa, enquanto quando em um *tablet* a uma mecânica de estratégia em tempo real.

A.74 XBOX Smart Glass

A tecnologia *Smart Glass*, presente no console de última geração *Xbox One* permite a integração de dispositivos móveis presentes no ambiente ao universo de jogo sendo vivido. Isso fica bem claro em jogos como *"Forza Horizon"* e *"Dead Rising 3"*.

A.74.1 Forza Horizon

Em *Forza Horizon*[17], o jogador faz parte de uma equipe de pilotos de carros participando de um grande festival de corrida no sul da Europa. Durante o jogo, diversos eventos ocorrem no universo virtual fornecendo ao usuário um mundo aberto contento diversas opções a serem usufruídas. Caso o jogador associe um tablet ao jogo, esse pode ser utilizado como mostrador do mapa de jogo, permitindo que a tela principal tenha seu foco na direção.

A.74.2 Dead Rising 3

O jogo *Dead Rising 3*[18] se passa na cidade ficcional de "*Los Perdidos*". O jogador interpreta o mecânico *Nick Ramos* que deve tentar se salvar da infestação zumbi que assola a cidade. É permitido ao usuário sincronizar seu *smartphone* com o console, passando a

[15] `http://www.pokemonbank.com/`
[16] `http://tomclancy-thedivision.ubi.com/game/en-US/home/`
[17] `http://www.forzamotorsport.net/`
[18] `https://store.xbox.com/pt-BR/Xbox-One/Games/Dead-Rising-3/affdf371-f512-4d8d-a9e9-dac4524a0e3a`

representar o dispositivo pertencente ao personagem principal. Durante a sessão, o telefone é utilizado por outros personagens para convidar o usuário a tomar parte em missões especiais. Desta forma, novas tramas são habilitadas, sendo estas não acessíveis aqueles que não façam uso deste recurso.

A.75 Remote Play

O recurso de *Remote Play* disponível entre o console *Play Station 4* e o portátil *PS Vita* permite que sessões de jogo iniciadas ainda na televisão sejam transportadas de forma transparente. Para que isto seja possível, o jogo deve ser adaptado para que seus comandos sejam tanto compatíveis com os controles do console como do portátil. Este é o caso do jogo *"God of War"* [19].

A.76 EVE: DUST 514

Dust 514 [20] se passa no mesmo universo de *Eve Online*[21]. Enquanto o último consiste em um jogo de tiro em primeira pessoa, o primeiro é um simulador de voo entre caças interplanetários. De toda forma, ambos os jogos ocorrem no mesmo universo (um *MMO*), sendo que as ações de um jogador influenciam nas do outro.

A.77 Battlefield Commander

O jogo de tiro em primeira pessoa "*Battlefield 4*™ " conta com um modo contendo uma mecânica de estratégia em tempo real denominado *Commander*. Jogadores no console podem optar por este modo, fornecendo suporte a seus companheiros que estão no campo de batalha. Porém, jogadores que possuam um tablet podem fazer uso de um aplicativo[22] que permite que esse modo seja experimentado em paralelo à partida no console.

A.78 SuperBetter

SuperBetter[23] criado pela *designer* Jane McGonigal, tem por objetivo auxiliar as pessoas a superarem desafios pessoais de forma colaborativa. Nesta plataforma, o usuário pode criar suas próprias missões e convidar amigos a serem seus aliados e ajudarem a completá-las. Conforme as missões são conquistadas, bonificações são adquiridas para incentivar que se continue.

[19] `https://godofwar.playstation.com`
[20] `http://dust514.com/`
[21] `http://www.eveonline.com/`
[22] `https://play.google.com/store/apps/details?id=com.ea.game.warsawcommander_row`
[23] `https://www.superbetter.com/`

A.79 Zombies Run

Este jogo[24] simula um mundo pós apocalíptico onde zumbis dominaram a face da terra. Para tomar parte do mundo do jogo, o usuário utiliza seu *smartphone* durante corridas pelas ruas da cidade, simulando que a saída de seu esconderijo. Usando, então, o retorno auditivo, o jogo provê *feedback* sobre hordas de zumbis que estejam atacando ou locais com suprimentos a serem coletados.

A.80 Run an Empire

Este jogo[25] desafia o jogador a conquistar o maior território possível. Para fazê-lo, o jogador deve desenhar perímetros através do percurso detectado pelo *GPS* de seu celular. Quanto mais vezes um jogador percorrer uma região, mais difícil de outro jogador tomá-la. Cabe então ao jogador equilibrar a conquista de novos territórios com a manutenção dos já existentes.

A.81 Space Team

Em *Space Team* [26] cada jogador tem o controle de um conjunto de controles e sensores complementares de uma nave viajando pela vastidão do espaço. Cabe ao grupo de cada sessão escolher como colaborar para trocar informações e garantir que as ações corretas sejam tomadas para que a nave mantenha-se em sua rota. O jogo se utiliza da descoberta de dispositivos na mesma rede sem fio para determinar quem faz parte da tripulação. Tornando a partida possível apenas àqueles que estão no mesmo local (ou muito próximos).

A.82 Moff

Este bracelete [27] reconhece os movimentos de acordo com padrões pré-estabelecidos. Cada tipo de gesto reproduz um conjunto de sons distintos. Usando estes efeitos sonoros, diversas brincadeiras infantis podem ser incrementadas com mais uma dimensão de criatividade.

A.83 CleverPet

CleverPet[28] é um dispositivo para interação e entretenimento canino. Usando pedais luminosos e um estoque de ração, o cachorro é desafiado a aprender uma sequência de comandos. Quando os comandos são executados com sucesso, um prêmio (alimento) é fornecido criando um laço de reforço positivo.

[24] `https://www.zombiesrungame.com/`
[25] `http://runanempire.com/`
[26] `http://www.sleepingbeastgames.com/spaceteam/`
[27] `http://www.moff.mobi/`
[28] `http://getcleverpet.com/`

A.84 RoomAlive

O projeto *RoomAlive* [53] é uma evolução do *IllumiRoom* [54] cuja intenção é trazer um alto grau de imersão para o entretenimento na sala de estar. Utilizando-se de diversos projetores e sensores *Kinect*, este sistema permite não apenas projetar as interações do sistema no ambiente como também reconhecer os movimentos do usuário de acordo.

A.85 Hackaball

A *Hackaball* [29] consiste em uma bola dotada de diversos recursos. Dentre eles temos uma variedade de sensores de movimento e orientação, e atuadores tanto hápticos como visuais (através de LEDs multi-coloridos). Em associação com um aplicativo móvel, esta bola permite que crianças criem suas próprias brincadeiras e jogos definindo regras baseadas nos recursos fornecidos pela bola.

A.86 TripBook

Este *e-book* [30] altera os elementos da história de acordo com a localização de seu leitor, no momento da leitura. Locais importantes da trama estão relacionados a cidades do país indetificado através de um sensor de GPS. Quando o leitor vai de um país a outro, estes locais são alterados dinâmicamente, correspondendo a pontos do país de destino. Desta forma, a trama muda sua linha geográfica acompanhando as viagens de seu leitor.

[29] `http://www.hackaball.com/us`
[30] `http://tripbooksmiles.com.br/`

Apêndice B

Testes realizados na plataforma uOS

Aqui estão descritos os testes realizados para a avaliação quantitativa do uso da plataforma *uOS*. Estes foram realizados em duas partes. A primeira compreendendo a avaliação apenas do tempo de atraso introduzido pela plataforma, sem a influencia da rede. Já na segunda foi considerado o tempo de entrega de uma mensagem entre dois nós em uma rede sem fio *802.11g*.

B.1 Atraso da Plataforma

Para se avaliar o atraso introduzido pela plataforma foi construído um código que exercita uma instância do *middleware*, realizando chamadas de serviço locais. Desta forma a influência da rede é eliminada da análise. O tempo analisado foi obtido através da diferença entre o tempo de relógio no momento da realização de uma chamada síncrona e o tempo de relógio obtido na recepção da mensagem. Tal tempo foi obtido através da chamada de sistema "*System.nanoTime()*" que permite obter o tempo com precisão de nanosegundos presente no relógio da máquina. O código responsável por executar esta lógica está apresentado nas Listagens B.1 e B.2.

```
1 private static void runTests(UOS uos, ArrayList<Long> data,
2     Call call)
3     throws ServiceCallException {
4   for(int i = 0 ; i < NUM_ROUNDS ; i++){
5     data.add(System.nanoTime());
6     uos.getGateway().callService(null, call);
7   }
8 }
```

Listing B.1: Código que realiza a chamada de serviço.

```
1 public void collectService(Call call, Response response,
2   CallContext ctx){
3   data.add(System.nanoTime());
4 }
```

Listing B.2: Código que realiza a coleta do tempo de chegada da chamada.

Cada chamada foi realizada um total de 1024 vezes[1]. Como os testes foram realizadas em plataformas não dedicadas existe grande influência de outras aplicações sendo executadas concorrentemente, desta forma para se obter um resultado consistente (reproduzível) foram eliminados das amostras os valores que se encontravam destoantes acima do desvio padrão da média obtida. Tais testes foram realizados em dez configurações de hardware distintas incluindo tanto computadores pessoais (portáteis e de mesa) como dispositivos móveis. Os resultados se encontram na Tabela B.1.

B.2 Atraso Total

Para se avaliar o tempo de atraso incluindo a rede foi utilizado novamente uma chamada síncrona onde foi avaliado o tempo do envio ao recebimento de sua resposta. Desta forma é possível avaliar o tempo total decorrido entre a plataforma construir uma requisição, seu envio pela rede, seu processamento pelo *driver* remoto, o retorno da resposta pela rede e consequente avaliação pela plataforma. As Listagens B.3 e B.4 apresentam como o código de envio e recebimento foram construídos, respectivamente. Como o tempo obtido representa tanto o envio como o recebimento, foi considerado como influência a metade do valor observado.

```
1  private static void runTests(UOS uos, ArrayList<Long> data,
2      Call call, UpDevice selectedDevice)
3      throws ServiceCallException {
4    for(int i = 0 ; i < NUM_ROUNDS ; i++){
5          long before = System.nanoTime();
6          gateway.callService(selectedDevice, call);
7          long after = System.nanoTime();
8          sum += after-before;
9          data.add(after-before);
10   }
11 }
```

Listing B.3: Código que realiza a chamada de serviço.

```
1  public void print(Call call, Response response,
2      CallContext ctx) {
3    String message = call.getParameterString("text");
4    message = String.format("%s: %s",
5        ctx.getCallerDevice().getName(), message);
6    System.out.println(message);
7    response.addParameter("echo", message);
8  }
```

Listing B.4: Código que realiza a coleta do tempo de chegada da chamada.

Cada chamada foi realizada um total de 1024 vezes, de forma similar ao realizado no teste de atraso da plataforma. Como a rede utilizada não era de uso exclusivo aos testes

[1]Representada pela constante NUM_ROUNDS

existe a presença de flutuações decorridas de outras aplicações e dispositivos presentes. A fim de se obter um resultado consistente foram novamente eliminados das amostras os valores que se encontravam destoantes acima do desvio padrão da média obtida. Estes testes foram realizados em diversas combinações ponto a ponto entre as configurações de *hardware* presentes e os dados obtidos podem ser observados na tabelaB.2.

Tempo de Atraso	Configuração
Média: 0,880 ms Mínimo: 0,778 ms Máximo: 0,906 ms	HTC Nexus One Cyanogen Mod 7.2.0 - Android 2.3.7 CPU 1 GHz Scorpion 512 MB RAM
Média: 0,516 ms Mínimo: 0,487 ms Máximo: 0,553 ms	LG Nexus 4 Android Lolipop 5.0 CPU 1.5 GHz Krait Quad-Core 2 GB RAM
Média: 0,212 ms Mínimo: 0,171 ms Máximo: 0,246 ms	Samsung Nexus 10 Android Lolipop 5.0 CPU 1.7 GHz Cortex-A15 Dual-Core 2 GB RAM
Média: 0,031 ms Mínimo: 0,026 ms Máximo: 0,034 ms	Dell Vostro 1500[1] Ubuntu 14.04 LTS 64 Bits CPU 2.20GHz Intel(R) Core 2 Duo 4 GB DDR2 333MHz RAM
Média: 0,026 ms Mínimo: 0,024 ms Máximo: 0,044 ms	Apple iMac - Windows Windows 7 Ultimate SP1 64 Bits CPU 3.06 Ghz Intel Core 2 Duo 4 GB 1.067 Mhz DDR2 RAM
Média: 0,018 ms Mínimo: 0,018 ms Máximo: 0,019 ms	Apple iMac - 1 Mac OSx 10.6.8 CPU 2.4 GHz Intel Core 2 Duo 4 Gb 800 Mhz DDR2 SDRAM
Média: 0,018 ms Mínimo: 0,014 ms Máximo: 0,023 ms	Sony Vaio ALlInOne Ubuntu 11.10 32 bits CPU 2.53 GHz Intel Core 2 Duo 8 GB RAM
Média: 0,015 ms Mínimo: 0,015 ms Máximo: 0,016 ms	Apple iMac - 2 MACOSX 10.6.8 CPU 3.06 Ghz Intel Core 2 Duo 4 GB 1.067 Mhz DDR2 RAM
Média: 0,010 ms Mínimo: 0,009 ms Máximo: 0,019 ms	Dell XPS 15 Ubuntu 14.04 LTS 64 Bits CPU 2.2 GHz Intel Core i7 Quad-Core 16 GB RAM
Média: 0,007 ms Mínimo: 0,006 ms Máximo: 0,008 ms	Sentey PC Ubuntu 14.04 LTS 64 Bits CPU 4.0 Ghz AMD Eight Core 8 GB RAM

Tabela B.1: Resultados do tempo de atraso da plataforma em diversas configurações distintas.

[1] : Esta é a mesma configuração utilizada nos testes originais do *middleware uOS*.

Origem	Destino	Tempo Médio	Tempo Mínimo	Tempo Máximo
Vaio AllInOne	XPS	50,374 ms	50,262 ms	200,529 ms
XPS	Apple iMac - 1	50,473 ms	50,276 ms	300,597 ms
iMac - 2	XPS 15	50,485 ms	50,142 ms	1200,918 ms
Vostro 1500	XPS 15	50,637 ms	50,214 ms	301,264 ms
XPS 15	Nexus One	50,850 ms	50,412 ms	201,012 ms
XPS 15	iMac - 1	50,864 ms	50,416 ms	702,171 ms
XPS 15	iMax - 2	50,866 ms	50,453 ms	451,627 ms
XPS 15	iMac Windows	50,872 ms	50,440 ms	201,326 ms
XPS 15	Vostro 1500	51,042 ms	50,399 ms	102,899 ms
XPS 15	Nexus 4	51,094 ms	50,412 ms	1152,744 ms
XPS 15	Vaio AllInOne	51,141 ms	50,435 ms	601,569 ms
XPS 15	Nexus 10	51,248 ms	50,446 ms	201,686 ms
Nexus One	XPS 15	55,043 ms	50,578 ms	1009,872 ms
Nexus 4	XPS 15	70,361 ms	55,218 ms	267,694 ms
Nexus 10	XPS 15	71,315 ms	54,209 ms	203,367 ms

Tabela B.2: Resultados do tempo de atraso utilizando uma rede sem fio para a comunicação.

Referências

[1] *Business Week*. Number Nº 3392-3405. Bloomberg L.P., 2012. URL `http://books.google.co.uk/books?id=kAseAQAAMAAJ`. 7

[2] F. Adelstein. *Fundamentals of mobile and pervasive computing*. McGraw-Hill professional engineering. McGraw-Hill, 2005. ISBN 9780071412377. URL `http://books.google.com.br/books?id=IhMfAQAAIAAJ`. 8

[3] E. Aitenbichler, J. Kangasharju, and M. Mühlhäuser. Mundocore: A light-weight infrastructure for pervasive computing. *Pervasive and Mobile Computing*, 3(4):332–361, 2007. URL `http://elara.tk.informatik.tu-darmstadt.de/publications/2007/aitenbichler07pmc.pdf`. 48

[4] O. Akribopoulos, D. Bousis, D. Efstathiou, H. Koutsouridis, M. Logaras, A. Loukas, A. Nafas, G. Oikonomou, I. Thireou, N. Vassilakis, P. Kokkinos, G. Mylonas, and I. Chatzigiannakis. A software platform for developing multi-player pervasive games using small programmable object technologies. In *Mobile Ad Hoc and Sensor Systems, 2008. MASS 2008. 5th IEEE International Conference on*, pages 544 –546, 29 2008-oct. 2 2008. doi: 10.1109/MAHSS.2008.4660084. 88

[5] The OSGi Alliance. *OSGi Service Platform Core Specification*. The OSGi Alliance, 2011. Diponível em `http://www.osgi.org/download/r4v43/r4.core.pdf` (acessado em 10/11/2014). 77

[6] Waern Annika and Denward Marie. On the edge of reality: Reality fiction in &#38#8216sanningen om marika&#38#8217. In *Breaking New Ground: Innovation in Games, Play, Practice and Theory*. Brunel University, September 2009. URL `http://www.digra.org/wp-content/uploads/digital-library/09287.50584.pdf`. 86

[7] Ivon Arroyo, Imran A. Zualkernan, and Beverly P. Woolf. Hoodies and barrels: Using a hide-and-seek ubiquitous game to teach mathematics. *Advanced Learning Technologies, IEEE International Conference on*, 0:295–299, 2011. doi: http://doi.ieeecomputersociety.org/10.1109/ICALT.2011.91. ix, 10, 11, 85

[8] Entertainment Software Association, Ipsos Insight, et al. *Essential facts about the computer and video game industry*. Entertainment Software Association, 2013. 7, 73

[9] Sriram Karthik Badam and Niklas Elmqvist. Polychrome: A cross-device framework for collaborative web visualization. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces*, ITS '14, pages 109–118, New York,

NY, USA, 2014. ACM. ISBN 978-1-4503-2587-5. doi: 10.1145/2669485.2669518. URL http://doi.acm.org/10.1145/2669485.2669518. 71

[10] Tilde Bekker, Janienke Sturm, and Berry Eggen. Designing playful interactions for social interaction and physical play. *Personal and Ubiquitous Computing*, 14 (5):385–396, 2010. ISSN 1617-4909. doi: 10.1007/s00779-009-0264-1. URL http://dx.doi.org/10.1007/s00779-009-0264-1. 13, 87

[11] P. Bellavista, A. Corradi, R. Montanari, and C. Stefanelli. Context-aware middleware for resource management in the wireless internet. *Software Engineering, IEEE Transactions on*, 29(12):1086 – 1099, dec. 2003. ISSN 0098-5589. doi: 10.1109/TSE.2003.1265523. 76

[12] P. Bellavista, A. Corradi, and E. Magistretti. Lightweight code mobility for proxy-based service rebinding in manet. In *Wireless Communication Systems, 2004, 1st International Symposium on*, pages 208 – 214, 2004. doi: 10.1109/ISWCS.2004.1407239. 76

[13] Steve Benford, Will Seager, Martin Flintham, Rob Anastasi, Duncan Rowland, Jan Humble, Danaë Stanton, John Bowers, Nick Tandavanitj, Matt Adams, Ju Farr, Amanda Oldroyd, and Jon Sutton. The error of our ways: The experience of self-reported position in a location-based game. In Nigel Davies, Elizabeth Mynatt, and Itiro Siio, editors, *UbiComp 2004: Ubiquitous Computing*, volume 3205 of *Lecture Notes in Computer Science*, pages 70–87. Springer Berlin / Heidelberg, 2004. ISBN 978-3-540-22955-1. ix, 9, 10, 12, 16, 93

[14] Steve Benford, Andy Crabtree, Martin Flintham, Adam Drozd, Rob Anastasi, Mark Paxton, Nick Tandavanitj, Matt Adams, and Ju Row-Farr. Can you see me now? *ACM Trans. Comput.-Hum. Interact.*, 13(1):100–133, March 2006. ISSN 1073-0516. doi: 10.1145/1143518.1143522. URL http://doi.acm.org/10.1145/1143518.1143522. 93

[15] S. Björk, J. Holopainen, P. Ljungstrand, and K. P. Akesson. Designing ubiquitous computing games - a report from a workshop exploring ubiquitous computing entertainment. *Personal Ubiquitous Comput.*, 6(5-6):443–458, 2002. 4, 8

[16] Elizabeth M. Bonsignore, Derek L. Hansen, Zachary O. Toups, Lennart E. Nacke, Anastasia Salter, and Wayne Lutters. Mixed reality games. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work Companion*, CSCW '12, pages 7–8, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1051-2. doi: 10.1145/2141512.2141517. URL http://doi.acm.org/10.1145/2141512.2141517. 8

[17] Isabella von Mühlen Brandalise. Áreas de riscos : a cidade em jogo. http://bdm.unb.br/handle/10483/7031, Fev 2014. Trabalho de conclusão de curso disponível em http://bdm.unb.br/handle/10483/7031 (acessado em 10/10/2014). 61

[18] Mark Weiser; John Seely Brow. Designing calm technology. Technical report, Xerox PARC, 1995. Diponível em `http://nano.xerox.com/weiser/calmtech/calmtech.htm` (acessado em 22/03/2012). 4

[19] Mat Buckland. *Programming Game AI By Example*. Jones & Bartlett Learning, Plano, TX, USA, 2004. ISBN 978-1556220784. 44

[20] Fabricio Nogueira Buzeto. Um conjunto de soluções para a construção de aplicativos de computação ubiqua. Master's thesis, Universidade de Brasília - UnB - Department of Computer Science, 2010. 71

[21] Fabricio Nogueira Buzeto, Carla Denise Castanho, and Ricardo Pezzuol Jacobi. up: A lightweight protocol for services in smart spaces. In *Proceedings of the 2011 Fourth International Conference on Ubi-Media Computing*, U-MEDIA '11, pages 25–30, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-0-7695-4493-9. doi: 10.1109/U-MEDIA.2011.47. URL `http://dx.doi.org/10.1109/U-MEDIA.2011.47`. 39

[22] Fabrício Buzeto, Carlos Filho, Carla Castanho, and Ricardo Jacobi. DSOA - A service Oriented Architecture for Ubiquitous Applications. *International Journal of Handheld Computing Research, pages 53-64*, 2011. 38

[23] F.N. Buzeto, M.A.M. Capretz, C.D. Castanho, and R.P. Jacobi. uos: A resource rerouting middleware for ubiquitous games. In *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*, pages 88–95, Dec 2013. doi: 10.1109/UIC-ATC.2013.44. 26

[24] A. Carzaniga, G.P. Picco, and G. Vigna. Is code still moving around? looking back at a decade of code mobility. In *Software Engineering - Companion, 2007. ICSE 2007 Companion. 29th International Conference on*, pages 9 –20, may 2007. doi: 10.1109/ICSECOMPANION.2007.44. 33, 48

[25] G. Castellano, I. Leite, A. Pereira, C. Martinho, A. Paiva, and P.W. McOwan. It's all in the game: Towards an affect sensitive and context aware game companion. In *Affective Computing and Intelligent Interaction and Workshops, 2009. ACII 2009. 3rd International Conference on*, pages 1 –8, sept. 2009. doi: 10.1109/ACII.2009.5349558. 86

[26] A.H.O.R. Castillo, F.N. Buzeto, C.D. Castanho, and R.P. Jacobi. An ontology-based model for context information management in smart spaces. In *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*, pages 278–284, Dec 2013. doi: 10.1109/UIC-ATC.2013.67. 78

[27] Adrian David Cheok, Xubo Yang, Zhou Zhi Ying, Mark Billinghurst, and Hirokazu Kato. Touch-space: Mixed reality game space based on ubiquitous, tangible, and social computing. *Personal Ubiquitous Comput.*, 6(5-6):430–442, January 2002. ISSN 1617-4909. doi: 10.1007/s007790200047. URL `http://dx.doi.org/10.1007/s007790200047`. 94

[28] Tanguy Coenen, Lien Mostmans, and Kris Naessens. Museus: Case study of a pervasive cultural heritage serious game. *J. Comput. Cult. Herit.*, 6(2):8:1–8:19, May 2013. ISSN 1556-4673. doi: 10.1145/2460376.2460379. URL `http://doi.acm.org/10.1145/2460376.2460379`. 80

[29] Rafael Simão da Costa. Run fast! : um jogo de corrida ubíquo dinamicamente recon?gurável. *UnB - Universidade de Brasília, Departamento de Ciências da Computação*, 2013. ix, 56, 57

[30] Chris Crawford. *The Art of Computer Game Design*. Osborne/McGraw-Hill, Berkeley, CA, USA, 1984. ISBN 0881341177. 18

[31] Cristiano André da Costa, Adenauer Corrêa Yamin, and Cláudio Fernando Resin Geyer. Toward a general software infrastructure for ubiquitous computing. *IEEE Pervasive Computing*, 7(1):64–73, 2008. ISSN 1536-1268. doi: http://doi.ieeecomputersociety.org/10.1109/MPRV.2008.21. 5, 18

[32] Leonardo G de Freitas, Luiggi Monteiro Reffatti, Igor Rafael de Sousa, Anderson C Cardoso, Carla Denise Castanho, Rodrigo Bonifácio, and Guilherme N Ramos. Gear2D: an extensible component-based game engine. In *Proceedings of the International Conference on the Foundations of Digital Games*, pages 81–88. ACM, 2012. 44

[33] Adam Drozd, Steve Benford, Nick Tandavanitj, Michael Wright, and Alan Chamberlain. Hitchers: Designing for cellular positioning. In Paul Dourish and Adrian Friday, editors, *UbiComp 2006: Ubiquitous Computing*, volume 4206 of *Lecture Notes in Computer Science*, pages 279–296. Springer Berlin / Heidelberg, 2006. ISBN 978-3-540-39634-5. URL `http://dx.doi.org/10.1007/11853565_17`. $10.1007/11853565_17.92$

[34] Douglas Easterly and Angela Blachnitzky. Ubiball: A ubiquitous computing game for children. In *Proceedings of the 13th International MindTrek Conference: Everyday Life in the Ubiquitous Era*, MindTrek '09, pages 41–44, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-633-5. doi: 10.1145/1621841.1621850. URL `http://doi.acm.org/10.1145/1621841.1621850`. 88

[35] W. Emmerich, C. Mascolo, and A. Finkelstein. Implementing incremental code migration with xml. In *Software Engineering, 2000. Proceedings of the 2000 International Conference on*, pages 397 –406, 2000. doi: 10.1109/ICSE.2000.870430. 76

[36] Elizabeth Evans, Martin Flintham, and Sarah Martindale. The malthusian paradox: performance in an alternate reality game. *Personal and Ubiquitous Computing*, 18(7): 1567–1582, 2014. ISSN 1617-4909. doi: 10.1007/s00779-014-0762-7. URL `http://dx.doi.org/10.1007/s00779-014-0762-7`. 16, 79

[37] P. Ferreira, J. Orvalho, and F. Boavida. A middleware architecture for mobile and pervasive large-scale augmented reality games. In *Fifth Annual Conference on Communication Networks and Services Research, 2007. CNSR '07.*, pages 203–212, 2007. doi: 10.1109/CNSR.2007.2. 71, 77

[38] Marisardo M Filho and Universidade Federal De Pernambuco. Narrativa Centrada no Jogador: Uma Análise da Relação entre a Narrativa Emergente e as Mecânicas nos Jogos Digitais. In *Brazilian Symposium on Computer Games and Digital Entertainment - SBGAMES*, pages 175–184, 2014. URL `http://www.sbgames.org/sbgames2014/files/papers/art_design/full/A&D_Full_NarrativaCentradanoJogador.pdf`. 5, 32, 48

[39] P.C. Garrido, G.M. Miraz, I.L. Ruiz, and M.A. Gomez-Nieto. Near field communication in the development of ubiquitous games. In *Internet Technology and Secured Transactions (ICITST), 2010 International Conference for*, pages 1 –7, nov. 2010. 86

[40] Chris Greenhalgh, Steve Benford, Adam Drozd, Martin Flintham, Alastair Hampshire, Leif Oppermann, Keir Smith, and Christoph von Tycowicz. Addressing mobile phone diversity in ubicomp experience development. In John Krumm, Gregory Abowd, Aruna Seneviratne, and Thomas Strang, editors, *UbiComp 2007: Ubiquitous Computing*, volume 4717 of *Lecture Notes in Computer Science*, pages 447–464. Springer Berlin / Heidelberg, 2007. ISBN 978-3-540-74852-6. 88, 89

[41] Jason Gregory. *Game Engine Architecture*. CRC Press, Boca Raton, FL, USA, 2009. ISBN 978-1-56881-413-1. 36

[42] Kaj Grønbæk, Ole S. Iversen, Karen Johanne Kortbek, Kaspar Rosengreen Nielsen, and Louise Aagaard. Igamefloor: A platform for co-located collaborative games. In *Proceedings of the International Conference on Advances in Computer Entertainment Technology*, ACE '07, pages 64–71, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-640-0. doi: 10.1145/1255047.1255061. URL `http://doi.acm.org/10.1145/1255047.1255061`. 89

[43] Bin Guo, Ryota Fujimura, Daqing Zhang, and Michita Imai. Design-in-play: improving the variability of indoor pervasive games. *Multimedia Tools and Applications*, 59(1):259–277, 2012. ISSN 1380-7501. doi: 10.1007/s11042-010-0711-z. URL `http://dx.doi.org/10.1007/s11042-010-0711-z`. 29, 30, 82

[44] Hong Guo, H. Trtteberg, AI Wang, and Meng Zhu. Temps: A conceptual framework for pervasive and social games. In *Digital Game and Intelligent Toy Enhanced Learning (DIGITEL), 2010 Third IEEE International Conference on*, April 2010. 9, 12, 14

[45] L. Gutierrez, I. Nikolaidis, E. Stroulia, S. Gouglas, G. Rockwell, P. Boechler, M. Carbonaro, and S. King. far-play: A framework to develop augmented/alternate reality games. In *IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011*, pages 531–536, 2011. doi: 10.1109/PERCOMW.2011.5766947. 20, 85

[46] Lucio Gutierrez, Eleni Stroulia, and Ioanis Nikolaidis. faars: A platform for location-aware trans-reality games. In Marc Herrlich, Rainer Malaka, and Maic Masuch, editors, *Entertainment Computing - ICEC 2012*, volume 7522 of *Lecture Notes in Computer Science*, pages 185–192. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-33541-9. doi: 10.1007/978-3-642-33542-6_16. URL `http://dx.doi.org/10.1007/978-3-642-33542-6_16`. 8, 20, 81

[47] Robert Headon and Rupert Curwen. Movement awareness for ubiquitous game control. *Personal and Ubiquitous Computing*, 6:407–415, 2002. ISSN 1617-4909. URL http://dx.doi.org/10.1007/s007790200045. 10.1007/s007790200045. 94

[48] Dominik Heckmann, Tim Schwartz, Boris Brandherm, Michael Schmitz, and Margeritta von Wilamowitz-Moellendorff. Gumo ? the general user model ontology. In Liliana Ardissono, Paul Brna, and Antonija Mitrovic, editors, *User Modeling 2005*, volume 3538 of *Lecture Notes in Computer Science*, pages 428–432. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-27885-6. doi: 10.1007/11527886_58. URL http://dx.doi.org/10.1007/11527886_58. 33

[49] C. Hiedels, C. Hoff, S. Rothkugel, and U. Wehling. Ubisettlers–a dynamically adapting mobile p2p multiplayer game for hybrid networks. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*, pages 109–113, March 2007. doi: 10.1109/PERCOMW.2007.120. 90

[50] J. Huizinga. *Homo Ludens: A Study of the Play-element in Culture*. Beacon paperbacks ; 15 : Sociology. Beacon Press, 1955. ISBN 9780807046814. URL http://books.google.com.br/books?id=oZgA8UDf3_4C. 13

[51] I. Hwang, Y. Lee, T. Park, and J. Song. Toward a mobile platform for pervasive games. In *Proceedings of the first ACM international workshop on Mobile gaming*, MobileGames '12, pages 19–24, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1487-9. doi: 10.1145/2342480.2342486. URL http://doi.acm.org/10.1145/2342480.2342486. 20, 77, 81

[52] R. Ierusalimschy. *Programming in Lua, Third Edition*. Roberto Ierusalimschy, 2012. ISBN 9788590379850. 49

[53] Brett Jones, Rajinder Sodhi, Michael Murdock, Ravish Mehra, Hrvoje Benko, Andrew Wilson, Eyal Ofek, Blair MacIntyre, Nikunj Raghuvanshi, and Lior Shapira. Roomalive: Magical experiences enabled by scalable, adaptive projector-camera units. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 637–644, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3069-5. doi: 10.1145/2642918.2647383. URL http://doi.acm.org/10.1145/2642918.2647383. 101

[54] Brett R. Jones, Hrvoje Benko, Eyal Ofek, and Andrew D. Wilson. Illumiroom: Peripheral projected illusions for interactive experiences. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 869–878, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1899-0. doi: 10.1145/2470654.2466112. URL http://doi.acm.org/10.1145/2470654.2466112. 101

[55] Ben Kirman, Conor Linehan, and Shaun Lawson. Blowtooth: a provocative pervasive game for smuggling virtual drugs through real airport security. *Personal and Ubiquitous Computing*, 16(6):767–775, 2012. ISSN 1617-4909. doi: 10.1007/s00779-011-0423-z. URL http://dx.doi.org/10.1007/s00779-011-0423-z. 86

[56] K. Koskinen and R. Suomela. Rapid prototyping of context-aware games. In *Intelligent Environments, 2006. IE 06. 2nd IET International Conference on*, volume 1, pages 135 –142, july 2006. 8, 20, 77, 90, 91

[57] D. Linner, F. Kirsch, I. Radusch, and S. Steglich. Context-aware multimedia provisioning for pervasive games. In *Multimedia, Seventh IEEE International Symposium on*, page 9 pp., dec. 2005. doi: 10.1109/ISM.2005.46. 92

[58] Robert Lübke, Daniel Schuster, and Alexander Schill. A framework for the development of mobile social software on android. In JoyYing Zhang, Jarek Wilkiewicz, and Ani Nahapetian, editors, *Mobile Computing, Applications, and Services*, volume 95 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 207–225. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-32319-5. doi: 10.1007/978-3-642-32320-1_14. URL `http://dx.doi.org/10.1007/978-3-642-32320-1_14`. 19, 84

[59] Andrew Macvean and Judy Robertson. Understanding exergame users' physical activity, motivation and behavior over time. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 1251–1260, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1899-0. doi: 10.1145/2470654.2466163. URL `http://doi.acm.org/10.1145/2470654.2466163`. 81

[60] D. Marples and P. Kriens. The open services gateway initiative: An introductory overview. *IEEE Communications Magazine*, 39(12):110–114. 77

[61] Jane Evelyn Mcgonigal. *This might be a game: ubiquitous play and performance at the turn of the twenty-first century*. PhD thesis, University of California, 2006. 12

[62] Victoria Meza-Kubo and AlbertoL. Morán. Ucsa: a design framework for usable cognitive systems for the worried-well. *Personal and Ubiquitous Computing*, 17(6):1135–1145, 2013. ISSN 1617-4909. doi: 10.1007/s00779-012-0554-x. URL `http://dx.doi.org/10.1007/s00779-012-0554-x`. 84

[63] Markus Montola. A ludological view on the pervasive mixed-reality game research paradigm. *Personal and Ubiquitous Computing*, 15(1):3–12, 2011. ISSN 1617-4909. doi: 10.1007/s00779-010-0307-7. URL `http://dx.doi.org/10.1007/s00779-010-0307-7`. 5, 17, 73

[64] Luca Mottola, Amy L. Murphy, and Gian Pietro Picco. Pervasive games in a mote-enabled virtual world using tuple space middleware. In *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, NetGames '06, New York, NY, USA, 2006. ACM. ISBN 1-59593-589-4. doi: 10.1145/1230040.1230098. URL `http://doi.acm.org/10.1145/1230040.1230098`. 77, 91

[65] Leif Oppermann, Gregor Broll, Mauricio Capra, and Steve Benford. Extending authoring tools for location-aware applications with an infrastructure visualization layer. In Paul Dourish and Adrian Friday, editors, *UbiComp 2006: Ubiquitous Computing*, volume 4206 of *Lecture Notes in Computer Science*, pages 52–68. Springer Berlin / Heidelberg, 2006. ISBN 978-3-540-39634-5. URL `http://dx.doi.org/10.1007/11853565_4`. $10.1007/11853565_4$. 92

[66] Rita Orji, Julita Vassileva, and ReganL. Mandryk. Lunchtime: a slow-casual game for long-term dietary behavior change. *Personal and Ubiquitous Computing*, 17(6):1211–1221,

2013. ISSN 1617-4909. doi: 10.1007/s00779-012-0590-6. URL http://dx.doi.org/10.1007/s00779-012-0590-6. 84

[67] Ana Paiva, Gerd Andersson, Kristina Höök, Dário Moura?o, Marco Costa, and Carlos Martinho. Sentoy in fantasya: Designing an affective sympathetic interface to a computer game. *Personal and Ubiquitous Computing*, 6:378–389, 2002. ISSN 1617-4909. URL http://dx.doi.org/10.1007/s007790200043. 10.1007/s007790200043. 93

[68] Lothar Pantel and Lars C. Wolf. On the impact of delay on real-time multiplayer games. In *Proceedings of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '02, pages 23–29, New York, NY, USA, 2002. ACM. ISBN 1-58113-512-2. doi: 10.1145/507670.507674. URL http://doi.acm.org/10.1145/507670.507674. 69

[69] M.C.S.C. Pimenta, F.N. Buzeto, L.H.O. Santos, C.D. Castanho, and R.P. Jacobi. A game engine for building ubigames. In *Network and Systems Support for Games (NetGames), 2014 13th Annual Workshop on*, pages 1–3, Dec 2014. doi: 10.1109/NetGames.2014.7008963. 40

[70] Rui Prada, Marco Vala, Ana Paiva, Kristina Hook, and Adrian Bullock. Fantasya ? the duel of emotions. In Thomas Rist, Ruth Aylett, Daniel Ballin, and Jeff Rickel, editors, *Intelligent Virtual Agents*, volume 2792 of *Lecture Notes in Computer Science*, pages 62–66. Springer Berlin / Heidelberg, 2003. ISBN 978-3-540-20003-1. URL http://dx.doi.org/10.1007/978-3-540-39396-2_11. 10.1007/978-3-540-39396-2$_1$1. 93

[71] Steve Rabin. *Introduction To Game Development, Second Edition*. Cengage Learning, Independence, KY, USA, 2010. ISBN 978-1584506799. 36

[72] Vaskar Raychoudhury, Jiannong Cao, Mohan Kumar, and Daqiang Zhang. Middleware for pervasive computing: A survey. *Pervasive and Mobile Computing*, 9(2):177 – 200, 2013. ISSN 1574-1192. doi: http://dx.doi.org/10.1016/j.pmcj.2012.08.006. URL http://www.sciencedirect.com/science/article/pii/S1574119212001113. Special Section: Mobile Interactions with the Real World. 18, 19

[73] Andrew Rollings and Dave Morris. *Game Architecture and Design: A New Edition*. New Riders Games, 2003. ISBN 0735713634. 18

[74] Kay Römer and Svetlana Domnitcheva. Smart playing cards: A ubiquitous computing game. *Personal and Ubiquitous Computing*, 6:371–377, 2002. ISSN 1617-4909. URL http://dx.doi.org/10.1007/s007790200042. 10.1007/s007790200042. 94

[75] Mizuki Sakamoto, Todorka Alexandrova, and Tatsuo Nakajima. Augmenting remote trading card play with virtual characters used in animation and game stories-towards persuasive and ambient transmedia storytelling. In *ACHI 2013, The Sixth International Conference on Advances in Computer-Human Interactions*, pages 168–177, 2013. 80

[76] Luciano Santos, Carla Castanho, Fabricio Buzeto, and Lucas Carvalho. A game engine plugin for ubigames development. In *SBGames 2014, Trilha de Computação*, 2014. 47

[77] Walt Scacchi. Computer game mods, modders, modding, and the mod scene. *First Monday*, 15(5), 2010. ISSN 13960466. URL `http://journals.uic.edu/ojs/index.php/fm/article/view/2965`. 32

[78] J. Schell. *The Art of Game Design: A book of lenses*. Morgan Kaufmann. Taylor & Francis, 2008. ISBN 9780123694966. URL `http://books.google.com.br/books?id=LP5xOYMjQKQC`. ix, 23, 24

[79] D. Schuster, D. Kiefner, R. Lubke, T. Springer, P. Bihler, and Holger Mugge. Step by step vs. catch me if you can: On the benefit of rounds in location-based games. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*, pages 155–160, March 2012. doi: 10.1109/PerComW.2012.6197469. 19, 83

[80] J.M. Silva and A. El Saddik. An adaptive game-based exercising framework. In *Virtual Environments Human-Computer Interfaces and Measurement Systems (VECIMS), 2011 IEEE International Conference on*, pages 1 –6, sept. 2011. doi: 10.1109/VECIMS.2011.6053847. 85

[81] C. Stach and L.F.M. Schlindwein. Candy castle - a prototype for pervasive health games. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*, pages 501–503, March 2012. doi: 10.1109/PerComW.2012.6197547. 82

[82] Christoph Stach. Gamework-a customizable framework for pervasive games. 2010. 21, 83

[83] Jaakko Stenros, Annika Waern, and Markus Montola. Studying the elusive experience in pervasive games. *Simul. Gaming*, 43(3):339–355, June 2012. ISSN 1046-8781. doi: 10.1177/1046878111422532. 4, 17

[84] Yoshiro Sugano, Jumpei Ohtsuji, Toshiya Usui, Yuya Mochizuki, and Naohito Okude. Shootball: The tangible ball sport in ubiquitous computing. In *Proceedings of the 2006 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, ACE '06, New York, NY, USA, 2006. ACM. ISBN 1-59593-380-8. doi: 10.1145/1178823.1178862. URL `http://doi.acm.org/10.1145/1178823.1178862`. 92

[85] K. Takashio, G. Soeda, and H. Tokuda. A mobile agent framework for follow-me applications in ubiquitous computing environment. In *Distributed Computing Systems Workshop, 2001 International Conference on*, pages 202 –207, apr 2001. doi: 10.1109/CDCS.2001.918706. 76

[86] B. Thomas, B. Close, J. Donoghue, J. Squires, P. De Bondi, M. Morris, and W. Piekarski. Arquake: an outdoor/indoor augmented reality first person application. In *Wearable Computers, The Fourth International Symposium on*, pages 139–146, Oct 2000. doi: 10.1109/ISWC.2000.888480. 95

[87] J.P. Tutzschke and O. Zukunft. Frap: a framework for pervasive games. In *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, EICS '09, pages 133–142, New York, NY, USA, 2009. ACM. 8, 87

[88] ITU International Telecommunication Union. Itu - statistics, 2009. URL http://www.itu.int. Diponível em http://www.itu.int/ITU-D/ict/statistics/ (acessado em 10/10/2014). 73

[89] Luis Valente, Bruno Feijó, and Julio Cesar Leite. Features and checklists to assist in pervasive mobile game development. In *SBGames 2013 - Trilha de Computação*, 2013. 12, 15, 17, 80

[90] Luis Valente, Bruno Feijó, and JulioCesarSampaio doPrado Leite. Mapping quality requirements for pervasive mobile games. *Requirements Engineering*, pages 1–29, 2015. ISSN 0947-3602. doi: 10.1007/s00766-015-0238-y. URL http://dx.doi.org/10.1007/s00766-015-0238-y. 12

[91] Rob van Kranenburg. The internet of things. a critique of ambient technology and the all-seeing network of rfid. Technical report, Institute of Network Cultures, 2007. Diponível em http://networkcultures.org/wpmu/portal/publications/network-notebooks/the-internet-of-things/ (acessado em 22/03/2012). 8

[92] G. Vanderhulst and L. Trappeniers. Public wifi hotspots at your service. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*, pages 411–414, March 2012. doi: 10.1109/PerComW.2012.6197522. 19, 84

[93] Mark Weiser. The computer for the 21st century. *Scientific American*, 1991. 4

[94] Mark Weiser. The world is not a desktop. *ACM Interactions*, 1993. 4

[95] Mark Weiser and John Seely Brown. The coming age of calm technology, 1996. 7

[96] Chin-Hao Wu, Yuan-Tse Chang, and Yu-Chee Tseng. Multi-screen cyber-physical video game: An integration with body-area inertial sensor networks. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, pages 832–834, March 2010. doi: 10.1109/PERCOMW.2010.5470554. 22, 47

[97] Raphael Zender, Richard Metzler, and Ulrike Lucke. Freshup?a pervasive educational game for freshmen. *Pervasive and Mobile Computing*, 14(0):47 – 56, 2014. ISSN 1574-1192. doi: http://dx.doi.org/10.1016/j.pmcj.2013.09.003. URL http://www.sciencedirect.com/science/article/pii/S1574119213001168. Special Issue on Pervasive Education Special Issue on The Social Car: Socially-inspired Mechanisms for Future Mobility Services. 79

[98] Zengbin Zhang, D. Chu, Xiaomeng Chen, and T. Moscibroda. Swordfight: Exploring phone-to-phone motion games. *Pervasive Computing, IEEE*, 11(4):8–12, Oct 2012. ISSN 1536-1268. doi: 10.1109/MPRV.2012.78. 82